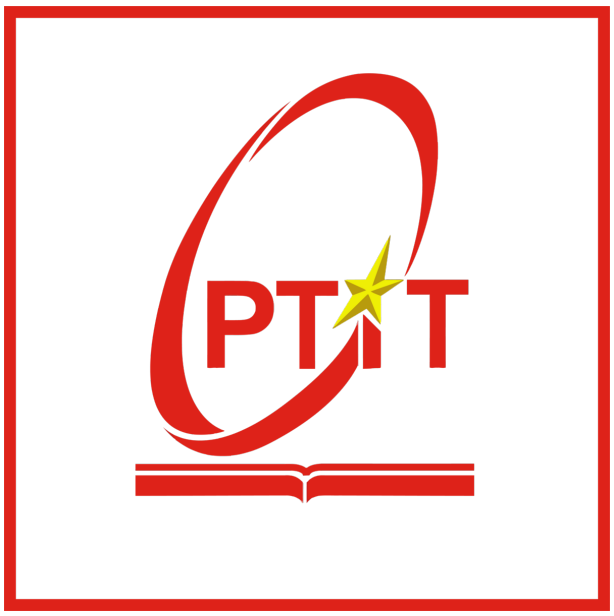
Học viện Công nghệ bưu chính viễn thông
KHOA CÔNG NGHỆ THÔNG TIN I



BÁO CÁO BÀI TẬP LỚN

NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn: Hoàng Kim Bách
Sinh viên: Đỗ Ngọc Huy
Mã sinh viên: B23DCCE045
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

MỞ ĐẦU

Báo cáo này trình bày các kết quả và phân tích chi tiết từ bài tập lớn môn Ngôn ngữ Lập trình Python, tập trung vào việc ứng dụng Python trong xử lý dữ liệu và Machine Learning trên dữ liệu thống kê cầu thủ Premier League. Báo cáo được cấu trúc thành các chương, mỗi chương đi sâu vào một khía cạnh cụ thể của quá trình phân tích:

- **Chương 1: Phân tích và Giải thích Mã Nguồn Thu thập Dữ liệu Thống kê Cầu thủ Premier League** tập trung vào quy trình thu thập, xử lý và tổng hợp dữ liệu thống kê chi tiết về các cầu thủ từ trang web fbref.com. Chương này mô tả cách sử dụng các thư viện như selenium, BeautifulSoup, và pandas để tương tác với trang web, trích xuất dữ liệu và chuẩn hóa chúng.
- **Chương 2: Báo cáo Giải thích Mã Phân tích Thống kê Bóng đá** trình bày việc làm sạch dữ liệu, tính toán các thống kê mô tả như trung bình, trung vị, độ lệch chuẩn ở cấp độ giải đấu và từng đội. Chương này cũng bao gồm việc xác định top 3 cầu thủ hàng đầu cho mỗi thống kê và trực quan hóa phân phối dữ liệu bằng biểu đồ histogram.
- **Chương 3: Phân tích và Phân cụm Dữ liệu Cầu thủ Bóng đá** đi sâu vào việc áp dụng thuật toán K-Means để nhóm các cầu thủ có đặc điểm thống kê tương đồng. Chương này mô tả quá trình chuẩn bị dữ liệu, xác định số lượng cụm tối ưu bằng phương pháp Elbow, và trực quan hóa kết quả phân cụm bằng PCA.
- **Chương 4: Phân Tích Dữ liệu Cầu thủ, Thu thập Giá trị Chuyển nhượng và Dự đoán Giá trị bằng Machine Learning tại Premier League** mô tả quy trình đa giai đoạn từ lọc cầu thủ, thu thập giá trị chuyển nhượng đã xác nhận và ước tính (ETV) từ web, đến việc xây dựng và huấn luyện mô hình Hồi quy Tuyến tính để dự đoán ETV dựa trên các chỉ số thống kê.

Mục lục

Chương 1

Phân tích và Giải thích Mã Nguồn Thu thập Dữ liệu Thống kê Cầu thủ Premier League

1.1 Giới thiệu

Chương này trình bày phân tích chi tiết về đoạn mã nguồn Python được phát triển nhằm mục đích thu thập, xử lý và tổng hợp dữ liệu thống kê chi tiết về các cầu thủ tham dự giải bóng đá Premier League mùa giải 2024-2025 từ trang web *fbref.com*. Đoạn mã sử dụng kết hợp các thư viện như *selenium* để tương tác với trang web động, *BeautifulSoup* để phân tích cú pháp HTML, *pandas* để xử lý và quản lý dữ liệu dạng bảng, và *webdriver_manager* để quản lý driver cho trình duyệt. Mục tiêu cuối cùng là tạo ra một tệp CSV chứa bộ dữ liệu thống kê toàn diện cho mỗi cầu thủ đủ điều kiện, phục vụ cho mục đích phân tích sâu hơn.

1.2 Công cụ và Thư viện Sử dụng

Đoạn mã sử dụng các thư viện Python sau:

- `time`: Dùng để tạm dừng quá trình thực thi của script, hữu ích khi chờ trang web tải xong nội dung.
- `pandas`: Thư viện mạnh mẽ cho thao tác và phân tích dữ liệu. Nó được sử dụng để đọc dữ liệu từ HTML vào DataFrame, xử lý dữ liệu (đổi tên cột, lọc, chuyển đổi kiểu dữ liệu), gộp các DataFrame và lưu kết quả ra CSV.
- `BeautifulSoup`: Thư viện dùng để phân tích cú pháp tài liệu HTML và XML. Trong script này, nó được dùng để tìm kiếm các bảng dữ liệu, đặc biệt là những bảng được đặt bên trong các comment HTML.
- `selenium`: Khung tự động hóa trình duyệt web. *FBref.com* là trang web sử dụng JavaScript để tải nội dung, do đó *selenium* là cần thiết để render trang đầy đủ trước khi trích xuất dữ liệu. Nó được sử dụng để mở URL và lấy nội dung HTML đã render.
- `selenium.webdriver.chrome.options.Options`: Dùng để cấu hình các tùy chọn cho trình duyệt Chrome, ví dụ chạy ở chế độ headless (ẩn giao diện).

- `selenium.webdriver.chrome.service.Service`: Dùng để chỉ định dịch vụ cho WebDriver (trong trường hợp này là Chrome).
- `webdriver_manager.chrome.ChromeDriverManager`: Thư viện giúp tự động tải xuống và quản lý ChromeDriver, loại bỏ nhu cầu tải xuống thủ công.
- `io.StringIO`: Dùng để xử lý chuỗi như một tệp tin, cho phép *pd.read_html* đọc dữ liệu HTML từ một chuỗi thay vì từ một tệp thực.
- `matplotlib.pyplot`: Thư viện dùng để tạo biểu đồ. (Lưu ý: Thư viện này được import trong mã nhưng không được sử dụng cho mục đích trực quan hóa dữ liệu trong đoạn mã được cung cấp. Nó có thể được dự định sử dụng trong các phần phân tích sau này.)
- `os`: Thư viện cung cấp cách tương tác với hệ điều hành, được sử dụng để tạo thư mục và xây dựng đường dẫn tệp.

1.3 Cấu trúc Mã Nguồn và Chức năng Chi tiết

Đoạn mã được tổ chức thành các phần chính:

1.3.1 Khởi tạo và Cấu hình Ban đầu

`C:\Users\84353\OneDrive\Desktop\BTL1_Python`: Định nghĩa đường dẫn thư mục gốc nơi mọi tệp đầu ra sẽ được lưu trữ. Việc sử dụng r""tạo ra một raw string, giúp tránh các vấn đề với ký tự escape "\".
Các hàm làm sạch và chuyển đổi dữ liệu:

- `convert_age_to_decimal(age_str)`: Chuyển đổi định dạng tuổi từ chuỗi (ví dụ: "25-180"hoặc "25.5") sang dạng số thập phân (ví dụ: 25.49 hoặc 25.50). Nó xử lý các trường hợp NaN hoặc "N/A"và làm tròn kết quả đến 2 chữ số thập phân. Hàm này rất quan trọng vì tuổi trên FBref có thể được biểu diễn dưới nhiều định dạng khác nhau.
- `extract_country_code(nation_str)`: Trích xuất mã quốc gia (ví dụ: "ENG", "FRA") từ chuỗi quốc tịch đầy đủ (ví dụ: "ENG England"). Nó lấy phần cuối cùng của chuỗi sau khi tách bằng khoảng trắng. Xử lý các trường hợp NaN hoặc "N/A".
- `clean_player_name(name)`: Làm sạch tên cầu thủ. Nó xử lý các trường hợp tên bị viết ngược (ví dụ: "Messi, Lionel") thành định dạng thông thường ("Lionel Messi") và loại bỏ khoảng trắng thừa. Xử lý các trường hợp NaN hoặc "N/A".

Thiết lập Selenium WebDriver:

- `options = Options()`: Tạo một đối tượng Options để tùy chỉnh cài đặt trình duyệt.
- `options.add_argument(-headless)`: Chạy trình duyệt ở chế độ ẩn (không hiển thị cửa sổ trình duyệt). Điều này hữu ích cho các script chạy tự động trên server hoặc khi không cần xem quá trình duyệt web.

- `options.add_argument(-disable-gpu), options.add_argument(-no-sandbox)`:Các tùy chọn bổ sung đôi khi cần thiết để WebDriver chạy ổn định trong các môi trường khác nhau.
- `driver = webdriver.Chrome(...)`: Khởi tạo trình duyệt Chrome. `Service(ChromeDriverManager().install())` đảm bảo ChromeDriver phù hợp được tải xuống và sử dụng tự động.

Định nghĩa Nguồn Dữ liệu:

- `urls`: Danh sách các URL trên fbref.com chứa các bảng thống kê khác nhau (chung, thủ môn, sút bóng, v.v.).
- `table_ids`: Danh sách các ID HTML tương ứng với các bảng dữ liệu cần trích xuất trên mỗi URL. Các ID này rất quan trọng để xác định đúng bảng.

Định nghĩa Cấu trúc Dữ liệu Đích:

- `required_columns`: Một danh sách chứa tên các cột mong muốn trong DataFrame kết quả cuối cùng. Thứ tự trong danh sách này cũng được sử dụng để sắp xếp lại các cột cuối cùng.
- `column_rename_dict`: Một từ điển lồng nhau. Khóa cấp 1 là ID của bảng ($table\_id$), và giá trị là một từ điển khác mapping tên cột gốc từ trang web (thường phức tạp hoặc không rõ ràng) với tên cột mong muốn được định nghĩa trong $required\_column$. Việc đổi tên này là cần thiết để chuẩn hóa tên các cột từ các bảng khác nhau trước khi gộp.

1.3.2 Thu thập và Xử lý Từng Bảng Dữ liệu

Đây là phần cốt lõi của việc scraping, được thực hiện trong vòng lặp `for url, table_id in zip(urls, table_ids):`.

- `driver.get(url)`: Mở URL hiện tại trong trình duyệt.
- `time.sleep(3)`: Dừng 3 giây để đảm bảo trang web có đủ thời gian tải nội dung, bao gồm cả các yếu tố được tải bằng JavaScript. Đây là một cách đơn giản nhưng đôi khi không hiệu quả hoàn toàn với các trang web phức tạp.
- `soup = BeautifulSoup(driver.page_source, "html.parser")`: Lấy nội dung HTML đầy đủ sau khi Selenium đã render trang và phân tích nó bằng BeautifulSoup.
- **Tìm bảng trong Comment HTML**: FBref.com thường đặt các bảng dữ liệu lớn bên trong các comment HTML để tránh tải chúng cùng lúc hoặc cho các mục đích khác.
  - `comments = soup.find_all(string=lambda text: isinstance(text, Comment))`: Tìm tất cả các comment trong tài liệu HTML.
  - Vòng lặp duyệt qua từng comment để tìm comment nào chứa ID của bảng ($if table\_id in comment$:).
  - Nếu tìm thấy, nội dung comment được phân tích lại bằng BeautifulSoup ($comment\_soup$) và bảng có ID tương ứng được tìm thấy (`table = comment_soup.find("table", {"id": table_id)`}).

- `df = pd.read_html(StringIO(str(table)), header=0)[0]`: Nếu tìm thấy bảng, nội dung HTML của bảng đó được chuyển thành chuỗi, sau đó được đọc vào DataFrame bằng *pd.read_html.StringIO* được dùng để đọc chuỗi HTML như một tệp. *header* = 0 chỉ định rằng hàng đầu tiên là header. [0] được thêm vào vì *read_html* trả về một danh sách các DataFrame (thường chỉ có một bảng).

- `df = df.rename(columns=column_rename_dict.get(table_id, {}))`: Đổi tên các cột của DataFrame hiện tại dựa trên mapping trong *column_rename_dict* cho *table_id* tương ứng. *.get*(*table_id*, {}) an toàn hơn, trả về một từ điển rỗng nếu *table_id* không có trong $column\_rename_dict$.

- `df = df.loc[:, df.columns.duplicated()]`: Xóa các cột có tên trùng lặp. Điều này xảy ra do cấu trúc header nhiều dòng của FBref, nơi các cột con có cùng tên với cột cha hoặc các cột khác có cùng tên sau khi đổi tên ban đầu.

- **Làm sạch cột 'Player' và 'Age' ban đầu**: Các hàm *clean_player_name* và *convert_age_to_decimal* được áp dụng cho các cột tương ứng ngay sau khi đọc và đổi tên DataFrame.

- `all_tables[table_id] = df`: Lưu DataFrame đã xử lý ban đầu vào từ điển *all_tables* với key là *table_id*.

1.3.3 Gộp và Xử lý Dữ liệu Tổng hợp

Sau khi thu thập và xử lý ban đầu cho từng bảng, các DataFrame riêng lẻ được gộp lại.

- `merged_df = None`: Khởi tạo DataFrame kết quả gộp.

- • Vòng lặp duyệt qua *all_tables*:

  - `df = df[[col for col in df.columns if col in required_columns]]`: Chỉ giữ lại các cột đã được liệt kê trong *required_columns* trong mỗi DataFrame trước khi gộp. Điều này giúp loại bỏ các cột không mong muốn và tránh các vấn đề khi gộp.
  - `df = df.drop_duplicates(subset=["Player"], keep="first")`: Xóa các hàng trùng lặp trong mỗi DataFrame dựa trên cột "Player". Điều này đảm bảo mỗi cầu thủ chỉ xuất hiện một lần trong mỗi bảng trước khi gộp.
  - Logic gộp:
    - Nếu*merged_df* còn *None* (lần đầu tiên), gán DataFrame hiện tại cho *merged_df*.
    - Nếu không, sử dụng *pd.merge*(*merged_df*, *df*, *on* = "*Player*", *how* = "*outer*", *validate* = "1 : 1").
      - `on="Player"`: Gộp dựa trên cột "Player".
      - `how="outer"`: Giữ tất cả các hàng từ cả hai DataFrame (merged_df hiện tại và df mới). Nếu một cầu thủ chỉ xuất hiện trong một bảng, các cột từ bảng kia sẽ có giá trị NaN.
      - `validate="1:1"`: Kiểm tra xem cột "Player"có phải là duy nhất trong cả hai DataFrames đang được gộp hay không. Nếu một cầu thủ xuất hiện nhiều lần trong một trong các DataFrame, sẽ gây ra lỗi. Lưu ý:

Với dữ liệu cầu thủ, *validate* = "1 : 1" có thể hơi chặt chẽ nếu có sự khác biệt nhỏ về tên hoặc ID ẩn, nhưng ở đây nó được sử dụng để xác nhận tính duy nhất của cầu thủ sau khi xử lý *drop_duplicates*.

- `merged_df = merged_df.loc[:, [col for col in required_columns if col in merged_df.columns]]`: Sắp xếp lại các cột trong DataFrame đã gộp theo đúng thứ tự được định nghĩa trong *required_columns*. Chỉ giữ lại những cột thực sự tồn tại trong *merged_df* sau khi gộp.
- **Chuyển đổi kiểu dữ liệu và làm sạch cuối cùng:**
  - `merged_df["Minutes"] = pd.to_numeric(merged_df["Minutes"], errors="coerce")`: Chuyển đổi cột "Minutes"sang kiểu số. *errors* = "*coerce*" sẽ thay thế bất kỳ giá trị nào không thể chuyển đổi thành số bằng NaN.
  - Các vòng lặp cho *int_columns* và *float_columns*: Chuyển đổi các cột số nguyên và số thực sang kiểu dữ liệu phù hợp. Tương tự, sử dụng *errors* = "*coerce*" để xử lý giá trị không hợp lệ thành NaN. Cột số nguyên được chuyển sang kiểu Int64 (kiểu số nguyên của pandas hỗ trợ NaN). Cột số thực được làm tròn đến 2 chữ số thập phân.
  - `merged_df = merged_df[merged_df["Minutes"].notna() & (merged_df["Minutes"] > 90)]`: Lọc bỏ những cầu thủ có giá trị "Minutes"là NaN hoặc thi đấu ít hơn 90 phút. Điều này thường được làm để tập trung vào các cầu thủ có đủ thời gian thi đấu để các thống kê "per 90"có ý nghĩa hơn.
  - `merged_df["Nation"] merged_df["Nation"].apply(extract_country_code)`: Áp dụng lại hàm *extract_country_code* cho cột "Nation". Có thể cần thiết sau khi gộp nếu cột này có giá trị từ nhiều nguồn khác nhau.
  - `merged_df["Player"] = merged_df["Player"].apply(clean_player_name)`: Áp dụng lại hàm *clean_player_name* cho cột "Player"để đảm bảo tính nhất quán sau khi gộp.
  - Điền giá trị NaN trong các cột chuỗi (*string_columns*) bằng "N/A"để biểu diễn rõ ràng các giá trị thiếu trong các cột văn bản.

1.3.4 Lưu Kết quả và Kết thúc

- `csv_dir = os.path.join(base_dir, "csv")`: Xây dựng đường dẫn đến thư mục "csv"bên trong thư mục gốc.
- `os.makedirs(csv_dir, exist_ok=True)`: Tạo thư mục "csv"nếu nó chưa tồn tại. *exist_ok* = *True* ngăn lỗi xảy ra nếu thư mục đã tồn tại.
- `result_path = os.path.join(csv_dir, "result.csv")`: Xây dựng đường dẫn đầy đủ cho tệp CSV đầu ra.
- `merged_df.to_csv(result_path, index=False, encoding="utf-8-sig", na_rep="N/A")`: Lưu DataFrame cuối cùng vào tệp CSV.
  - `index=False`: Không ghi index của DataFrame vào tệp CSV.

  - `encoding="utf-8-sig"`: Sử dụng mã hóa UTF-8 với Byte Order Mark (BOM), hữu ích khi mở tệp CSV trong các chương trình như Microsoft Excel để đảm bảo hiển thị đúng các ký tự đặc biệt.
  - `na_rep="N/A"`: Biểu diễn các giá trị NaN trong DataFrame bằng chuỗi "N/A"trong tệp CSV.

- `driver.quit()`: Đóng trình duyệt và kết thúc phiên Selenium. Đây là bước quan trọng để giải phóng tài nguyên hệ thống.

1.4 Quy trình Xử lý Dữ liệu Tổng thể

Quy trình xử lý dữ liệu của script có thể được tóm tắt như sau:

- **Thu thập Thô**: Sử dụng Selenium để truy cập từng URL thống kê.
- **Phân tích HTML**: Sử dụng BeautifulSoup để phân tích nội dung HTML đã render, đặc biệt tìm kiếm các bảng ẩn trong comment.
- **Đọc vào DataFrame**: Chuyển đổi HTML của bảng thành DataFrame pandas.
- **Chuẩn hóa Cột**: Đổi tên các cột khó hiểu hoặc không nhất quán sang tên chuẩn, loại bỏ cột trùng lặp.
- **Làm sạch Ban đầu**: Áp dụng các hàm làm sạch cơ bản cho cột "Player"và "Age"trong từng DataFrame.
- **Lưu trữ Tạm thời**: Lưu từng DataFrame đã xử lý ban đầu vào một từ điển.
- **Lọc Cột Quan Tâm**: Trước khi gộp, chỉ giữ lại các cột cần thiết trong mỗi DataFrame.
- **Gộp DataFrame**: Gộp tất cả các DataFrame riêng lẻ lại dựa trên cột "Player"bằng phép gộp ngoài (*outermerge*) để giữ lại tất cả các cầu thủ.
- **Sắp xếp Cột**: Sắp xếp lại thứ tự các cột trong DataFrame đã gộp.
- **Chuyển đổi Kiểu Dữ liệu**: Chuyển đổi các cột số sang kiểu dữ liệu phù hợp (số nguyên, số thực), xử lý giá trị không hợp lệ thành NaN.
- **Lọc Cầu thủ**: Giữ lại chỉ những cầu thủ có đủ số phút thi đấu.
- **Làm sạch Cuối cùng**: Áp dụng lại các hàm làm sạch cho "Nation"và "Player", điền giá trị thiếu cho các cột chuỗi.
- **Lưu trữ Kết quả**: Lưu DataFrame cuối cùng đã làm sạch và tổng hợp vào tệp CSV.

1.5 Nhận xét và Đánh giá

Đoạn mã thực hiện tốt mục tiêu thu thập và tổng hợp dữ liệu từ nhiều bảng trên fbref.com. Việc sử dụng Selenium là phù hợp để xử lý trang web động. Kỹ thuật tìm bảng trong comment HTML là một cách tiếp cận thông minh để đối phó với cấu trúc cụ thể của fbref.com. Việc sử dụng pandas giúp xử lý dữ liệu một cách hiệu quả.
Tuy nhiên, script cũng có một số điểm cần lưu ý:

- **Tính ổn định**: Việc sử dụng *time.sleep*() để chờ trang tải có thể không đáng tin cậy. Các phương pháp chờ thông minh hơn của Selenium (như Explicit Waits) sẽ tốt hơn.
- **Tính bền vững**: $FBref.com$ có thể thay đổi cấu trúc HTML, ID bảng, tên cột gốc hoặc cách ẩn bảng trong comment bất cứ lúc nào, điều này có thể làm hỏng script.
- **Hiệu suất**: Truy cập từng URL và chờ đợi có thể tốn thời gian khi xử lý lượng lớn dữ liệu hoặc nhiều mùa giải.
- **Xử lý lỗi**: Mặc dù có các khối *try...except* cơ bản, việc xử lý lỗi chi tiết hơn (ví dụ: xử lý lỗi mạng, lỗi tìm phần tử HTML cụ thể) có thể làm cho script mạnh mẽ hơn.
- **SCA/GCA**: Trong $column\_rename\_dict$, tên cột SCA và GCA cho bảng $stats\_gca$ không được định nghĩa lại, chỉ có $SCA.1$ (đổi thành $SCA90$) và $GCA.1$ (đổi thành $GCA90$). Các cột SCA và GCA gốc (tổng số) có thể sẽ bị bỏ qua nếu không có ánh xạ đổi tên hoặc không nằm trong $required\_columns$. Kiểm tra lại $required\_columns$ và $column\_rename\_dict$ cho bảng $stats\_gca$. (Sau khi kiểm tra code, các cột SCA và GC gốc thực sự không có trong $required\_columns$, do đó chúng sẽ bị loại bỏ trước khi gộp).

1.6 Kết luận

Đoạn mã Python đã cung cấp một giải pháp hiệu quả để thu thập, làm sạch và tổng hợp dữ liệu thống kê cầu thủ Premier League từ fbref.com vào một tệp CSV duy nhất. Nó sử dụng các thư viện mạnh mẽ và kỹ thuật phù hợp để xử lý tính chất động và cấu trúc HTML đặc biệt của trang web nguồn. Bộ dữ liệu thu thập được là một nền tảng tốt cho các phân tích thống kê chuyên sâu về hiệu suất cầu thủ. Cần xem xét các cải tiến về tính ổn định và khả năng chống lại thay đổi cấu trúc trang web trong tương lai.

Chương 2

Báo cáo Giải thích Mã Phân tích Thống kê Bóng đá

2.1 Cài đặt Ban đầu và Tải Dữ liệu

Phần đầu tiên của script thiết lập môi trường làm việc và tải dữ liệu đầu vào.

```
import time
import pandas as pd
from bs4 import BeautifulSoup, Comment
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from io import StringIO
import matplotlib.pyplot as plt
import os
# Định nghĩa thư mục gốc
base_dir = r"C:\Users\84353\OneDrive\Desktop\BTL1_Python
# Định nghĩa thư mục cho các tệp CSV đầu vào và đầu ra
csv_dir = os.path.join(base_dir, "csv")
# Định nghĩa đường dẫn đến tệp CSV đầu vào trong thư mục csv
input_csv_path = os.path.join(csv_dir, "result.csv") # Cập nhật đường dẫn
# Tạo thư mục 'csv' nếu nó chưa tồn tại (hữu ích khi chạy lần đầu)
os.makedirs(csv_dir, exist_ok=True)
print(f"Đảm bảo thư mục {csv_dir} tồn tại.")
# Đọc tệp CSV vào DataFrame của pandas
try:
    # na_values=["N/A"] để đảm bảo các giá trị "N/A"được đọc là NaN (Not a Number)
    df = pd.read_csv(input_csv_path, na_values=["N/A"])
    print(f"Tải dữ liệu thành công từ {input_csv_path}")
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy tệp đầu vào tại {input_csv_path}")
    exit() # Thoát nếu không tìm thấy tệp đầu vào
except Exception as e:
    print(f"Lỗi khi tải tệp CSV: {e}")
    exit() # Thoát nếu có lỗi khác khi tải
```

- **Import Thư viện**: Mã nhập các thư viện cần thiết như pandas để xử lý dữ liệu, *matplotlib.pyplot* để vẽ biểu đồ, *os* để tương tác với hệ điều hành (quản lý thư mục và đường dẫn), và các thư viện khác liên quan đến web scraping ($selenium, bs4, webdriver\_manager$) mặc dù phần web scraping không được sử dụng trực tiếp trong đoạn mã phân tích này.

- **Định nghĩa Thư mục**: Xác định thư mục gốc ($base\_dir$) và thư mục con cho các tệp CSV (csv_dir). Điều này giúp tổ chức các tệp đầu vào và đầu ra một cách có cấu trúc.

- `Kiểm tra và Tạo Thư mục`: Đảm bảo thư mục csv tồn tại. Nếu chưa có, nó sẽ được tạo ra.

- `Đọc CSV`: Cố gắng đọc tệp *result.csv* vào một DataFrame của pandas. $na\_values = ["N/A"]$ được sử dụng để đảm bảo các giá trị "N/A"trong tệp CSV được hiểu là giá trị thiếu (NaN). Mã bao gồm xử lý lỗi để bắt các trường hợp tệp không tồn tại hoặc có lỗi khi đọc.

2.2 Làm sạch và Chuẩn bị Dữ liệu

Sau khi tải dữ liệu, mã tiến hành làm sạch và chuẩn bị dữ liệu cho các phân tích tiếp theo.

```
# Tạo một bản sao của DataFrame để thực hiện các tính toán, chuyển NaN thành 0 ở các cột số df_calc = df.copy()
# Định nghĩa các cột cần loại trừ (không phải là số)
exclude_columns = ["Player", "Nation", "Team", "Position"]
# Chuyển NaN thành 0 trong các cột số để tính toán
numeric_columns = [col for col in df_calc.columns if col not in exclude_columns] for col in numeric_columns:
  # Chuyển sang dạng số, đảm bảo NaN cho các giá trị không phải số, sau đó điền 0 vào NaN
  # Sử dụng errors='coerce' để biến các giá trị không phải số thành NaN trước khi điền
  df_calc[col] = pd.to_numeric(df_calc[col], errors="coerce").fillna(0)
print("Dữ liệu đã được làm sạch và các cột số đã được xử lý.")
```

- **Tạo Bản sao**: Tạo một bản sao của DataFrame gốc (df_calc). Điều này quan trọng để tránh sửa đổi dữ liệu gốc và để thực hiện các phép tính mà không ảnh hưởng đến DataFrame ban đầu.

- **Xác định Cột Số**: Tạo danh sách $numeric\_columns$ chứa tên của tất cả các cột ngoại trừ các cột định danh hoặc phân loại như "Player", "Nation", "Team", "Position".

- **Xử lý Giá trị Thiếu (NaN)**: Lặp qua các cột số, chuyển đổi chúng sang kiểu dữ liệu số bằng $pd.to\_numeric$ (với $errors = 'coerce'$ để biến các giá trị không thể chuyển đổi thành NaN), và sau đó điền các giá trị NaN bằng 0 bằng $fillna(0)$. Việc này đảm bảo rằng các phép tính toán học có thể được thực hiện mà không gặp lỗi do giá trị thiếu.

2.3 Tạo Tệp *top_3.txt*

Phần này của mã xác định 3 cầu thủ hàng đầu và 3 cầu thủ có giá trị thấp nhất cho mỗi thống kê số và lưu kết quả vào một tệp văn bản.

```
# 1. Tạo tệp top_3.txt
rankings = {}
for col in numeric_columns:
    # Xử lý các trường hợp mà một cột có thể hoàn toàn là 0 hoặc không phải số sau khi chuyển đổi
    # Kiểm tra nếu tổng tất cả các giá trị là 0 nhưng vẫn có dữ liệu trong cột
    if df_calc[col].sum() == 0 and df_calc[col].count() > 0:
        print(f"Bỏ qua xếp hạng cho '{col}' vì tất cả các giá trị đều là 0.")
        continue # Bỏ qua cột này và chuyển sang cột tiếp theo nếu tất cả giá trị đều là 0
    # Top 3 Cao nhất
    # Sử dụng copy() để tránh SettingWithCopyWarning
    top_3_high = df_calc[["Player", "Team", col]].sort_values(by=col, ascending=False).head(3).copy()
    top_3_high = top_3_high.rename(columns={col: "Value"})
    top_3_high["Rank"] = ["1st", "2nd", "3rd"]
    # Top 3 Thấp nhất (chỉ xem xét các giá trị khác 0 nếu tồn tại)
    # Sử dụng copy()
    non_zero_df = df_calc[df_calc[col] > 0].copy()
    if not non_zero_df.empty:
        # Sử dụng copy()
        top_3_low = non_zero_df[["Player", "Team", col]].sort_values(by=col, ascending=True).head(3).copy()
    else:
        # Nếu không có giá trị khác 0, lấy 3 giá trị thấp nhất từ dữ liệu gốc (sẽ là 0)
        # Sử dụng copy()
        top_3_low = df_calc[["Player", "Team", col]].sort_values(by=col, ascending=True).head(3).copy()
    top_3_low = top_3_low.rename(columns={col: "Value"})
    top_3_low["Rank"] = ["1st", "2nd", "3rd"]
    rankings[col] = {
        "Highest": top_3_high,
        "Lowest": top_3_low
    }
```

```
# Lưu kết quả vào tệp top_3.txt trong base_dir
top_3_path = os.path.join(base_dir, "top_3.txt")
with open(top_3_path, "w", encoding="utf-8") as f:
    for stat, data in rankings.items():
    f.write(f"\nThống kê: {stat}\n")
    f.write("\nTop 3 Cao nhất:\n")
    # Đảm bảo các cột tồn tại trước khi cố gắng in
    if not data["Highest"].empty:
        f.write(data["Highest"][["Rank", "Player", "Team", "Value"]].to_string(index=False))
    else:
        f.write("Không có dữ liệu.\n")
    f.write("\n\nTop 3 Thấp nhất:\n")
    if not data["Lowest"].empty:
        f.write(data["Lowest"][["Rank", "Player", "Team", "Value"]].to_string(index=False))
    else:
        f.write("Không có dữ liệu.\n")
    f.write("\n"+ "* 50 + "\n")
print(f"Đã lưu xếp hạng top 3 vào top_3_path")
```

- **Lặp qua Thống kê**: Mã lặp qua từng cột trong *numeric_columns*.
- **Xác định Top 3 Cao nhất**: Đối với mỗi thống kê, nó sắp xếp DataFrame theo cột đó theo thứ tự giảm dần ($ascending = False$) và chọn 3 hàng đầu tiên ($head(3)$). Kết quả được lưu trong top_3_high.
- **Xác định Top 3 Thấp nhất**: Đối với các giá trị thấp nhất, mã tạo một DataFrame tạm thời chỉ chứa các giá trị lớn hơn 0. Nếu DataFrame này không rỗng, nó sẽ tìm 3 giá trị thấp nhất từ đó. Nếu rỗng (tất cả giá trị là 0 hoặc âm), nó sẽ lấy 3 giá trị thấp nhất từ DataFrame gốc (có thể bao gồm 0). Kết quả được lưu trong top_3_low.
- **Lưu vào Tệp**: Kết quả top 3 cho từng thống kê (cả cao nhất và thấp nhất) được định dạng và ghi vào tệp $top\_3.txt$ trong thư mục gốc.

2.4 Tính toán Thống kê Trung bình, Trung vị và Độ lệch chuẩn

Phần này tính toán các thống kê mô tả (trung vị, trung bình, độ lệch chuẩn) cho toàn bộ giải đấu và cho từng đội, sau đó lưu kết quả vào một tệp CSV mới.

```
# 2. Tính toán trung vị (median), trung bình (mean) và độ lệch chuẩn (standard deviation)
    cho tệp results2.csv
rows = []
# Thêm hàng tổng thể trước
all_stats = {"": "all"}
for col in numeric_columns:
  # Đảm bảo cột là số trước khi tính toán thống kê
  if pd.api.types.is_numeric_dtype(df_calc[col]):
      all_stats[f"Trung vị của {col}"] = df_calc[col].median()
      all_stats[f"Trung bình của {col}"] = df_calc[col].mean()
      all_stats[f"Độ lệch chuẩn của {col}"] = df_calc[col].std()
  else:
      all_stats[f"Trung vị của {col}"] = None # Hoặc một chỉ báo nào đó
      all_stats[f"Trung bình của {col}"] = None
      all_stats[f"Độ lệch chuẩn của {col}"] = None
rows.append(all_stats)
# Tính toán thống kê cho từng đội
teams = sorted(df_calc["Team"].unique()) # Lấy danh sách các đội duy nhất và sắp xếp
for team in teams:
  # Sử dụng copy()
  team_df = df_calc[df_calc["Team"] == team].copy()
  team_stats = {"": team}
  for col in numeric_columns:
      if pd.api.types.is_numeric_dtype(team_df[col]):
          team_stats[f"Trung vị của {col}"] = team_df[col].median()
          team_stats[f"Trung bình của {col}"] = team_df[col].mean()
          team_stats[f"Độ lệch chuẩn của {col}"] = team_df[col].std()
      else:
          team_stats[f"Trung vị của {col}"] = None
          team_stats[f"Trung bình của {col}"] = None
          team_stats[f"Độ lệch chuẩn của {col}"] = None
  rows.append(team_stats)
```

```
# Tạo DataFrame từ các hàng thống kê
results_df = pd.DataFrame(rows)
# Đổi tên cột đầu tiên
results_df = results_df.rename(columns={"": "Đội/Tổng thể"})
for col in results_df.columns:
  if col != "Đội/Tổng thể":
      # Chỉ làm tròn các cột số
      if pd.api.types.is_numeric_dtype(results_df[col]):
          results_df[col] = results_df[col].round(2)
# Lưu kết quả vào tệp results2.csv trong thư mục 'csv'
results2_path = os.path.join(csv_dir, "results2.csv")
results_df.to_csv(results2_path, index=False, encoding="utf-8-sig")
print(f"Đã lưu thống kê thành công vào {results2_path} với {results_df.shape[0]} hàng và
    {results_df.shape[1]} cột.")
```

- **Thống kê Toàn giải đấu**: Tính toán trung vị, trung bình và độ lệch chuẩn cho từng cột số trên toàn bộ DataFrame (df_calc). Kết quả được lưu vào một dictionary all_stats.
- **Thống kê theo Đội**: : Lặp qua từng đội duy nhất trong dữ liệu. Đối với mỗi đội, nó lọc DataFrame để chỉ chứa dữ liệu của đội đó ($team\_df$) và tính toán các thống kê tương tự (trung vị, trung bình, độ lệch chuẩn) cho từng cột số. Kết quả được lưu vào dictionary $team\_stats$.
- **Tạo và Lưu DataFrame**: Các dictionary chứa thống kê (tổng thể và theo đội) được thu thập vào một danh sách $rows$. Danh sách này sau đó được chuyển đổi thành một DataFrame ($results\_df$). Cột đầu tiên được đổi tên thành "Đội/Tổng thể". Các giá trị số được làm tròn đến 2 chữ số thập phân. Cuối cùng, DataFrame này được lưu vào tệp $results2.csv$ trong thư mục csv.

2.5 Vẽ Biểu đồ Histogram

Phần này tạo các biểu đồ histogram để trực quan hóa phân phối của một số thống kê đã chọn, cả ở cấp độ giải đấu và cấp độ đội.

```
# 3. Vẽ biểu đồ histogram cho các thống kê đã chọn
selected_stats = ["Gls per 90", "xG per 90", "SCA90", "GA90", "TklW", "Blocks"]
histograms_dir = os.path.join(base_dir, "histograms")
league_dir = os.path.join(histograms_dir, "league")
teams_dir = os.path.join(histograms_dir, "teams")
# Tạo các thư mục lưu histogram
os.makedirs(histograms_dir, exist_ok=True)
os.makedirs(league_dir, exist_ok=True)
os.makedirs(teams_dir, exist_ok=True)
print(f"Đảm bảo các thư mục {league_dir} và {teams_dir} tồn tại.")
# Lấy danh sách các đội đã sắp xếp
teams = sorted(df_calc["Team"].unique())
for stat in selected_stats:
  # Kiểm tra xem thống kê có tồn tại và là kiểu số hay không if stat not in df_calc.columns
    or not pd.api.types.is_numeric_dtype(df_calc[stat]):
     print(f"Thống kê '{stat}' không tìm thấy hoặc không phải là số trong DataFrame. Bỏ
    qua việc tạo histogram.")
     continue
  # Histogram toàn giải đấu
  plt.figure(figsize=(10, 6))
  plt.hist(df_calc[stat], bins=20, color="skyblue", edgecolor="black")
  plt.title(f"Phân phối toàn giải đấu của {stat}")
  plt.xlabel(stat)
  plt.ylabel("Số lượng cầu thủ")
  plt.grid(True, alpha=0.3)
  # Lưu biểu đồ
  plt.savefig(os.path.join(league_dir, f"{stat}_league.png"), bbox_inches="tight")
  plt.close() # Đóng biểu đồ để giải phóng bộ nhớ
  print(f"Đã lưu histogram toàn giải đấu cho {stat}")
  # Histogram cho từng đội
  for team in teams:
     # Sử dụng copy()
     team_data = df_calc[df_calc["Team"] == team].copy()
     # Kiểm tra nếu dữ liệu đội rỗng hoặc cột thống kê không phải số
     if team_data.empty or not pd.api.types.is_numeric_dtype(team_data[stat]):
          print(f"Bỏ qua histogram cho '{team}' - '{stat}' do dữ liệu rỗng hoặc cột không
    phải số.")
          continue
```

```
    plt.figure(figsize=(8, 6))
    # Sử dụng màu khác nhau cho các thống kê phòng ngự
    color = "lightgreen"if stat in ["GA90", "TklW", "Blocks"] else "skyblue"
    plt.hist(team_data[stat], bins=10, color=color, edgecolor="black", alpha=0.7)
    plt.title(f"{team} - Phân phối của {stat}")
    plt.xlabel(stat)
    plt.ylabel("Số lượng cầu thủ")
    plt.grid(True, alpha=0.3)
    # Thay thế khoảng trắng và dấu gạch chéo cho tên tệp stat_filename = stat.replace("",
    "_").replace("/", "_")
    # Lưu biểu đồ cho từng đội
            plt.savefig(os.path.join(teams_dir,          f"{team}_{stat_filename}.png"),
    bbox_inches="tight")
    plt.close() # Đóng biểu đồ
    print(f"Đã lưu histogram cho {team} - {stat}")
print("Tất cả các histogram cho các thống kê đã chọn đã được tạo và lưu trong thư mục
    'histograms'.")
```

- **Chọn Thống kê**: Một danh sách *selected_stats* được định nghĩa chứa tên của các thống kê mà bạn muốn tạo histogram.
- **Tạo Thư mục Histogram**: Tạo các thư mục để lưu trữ các biểu đồ: $histograms, histograms/league$, và $histograms/teams$.
- **Histogram Toàn giải đấu**: Lặp qua các thống kê đã chọn. Đối với mỗi thống kê, nó tạo một biểu đồ histogram bằng cách sử dụng dữ liệu từ toàn bộ giải đấu ($df\_calc[stat]$). Biểu đồ được tùy chỉnh với tiêu đề, nhãn trục và lưới. Sau đó, biểu đồ được lưu dưới dạng tệp PNG trong thư mục $histograms/league$.
- **Histogram theo Đội**: Đối với mỗi thống kê đã chọn, mã tiếp tục lặp qua từng đội. Nó lọc dữ liệu cho đội hiện tại ($team\_data$) và tạo một biểu đồ histogram cho thống kê đó chỉ sử dụng dữ liệu của đội. Màu sắc của histogram có thể thay đổi tùy thuộc vào loại thống kê (ví dụ: màu xanh lá cây cho thống kê phòng ngự). Biểu đồ được lưu dưới dạng tệp PNG trong thư mục $histograms/teams$, với tên tệp bao gồm tên đội và tên thống kê. $plt.close()$ được gọi sau mỗi lần lưu để giải phóng bộ nhớ.

2.6 Xác định Đội có Giá trị Trung bình Cao nhất cho Mỗi Thống kê

Phần này tính toán giá trị trung bình của mỗi thống kê số cho từng đội và xác định đội nào có giá trị trung bình cao nhất cho mỗi thống kê.

- **Tính Trung bình theo Đội**: Sử dụng phương thức $groupby("Team")$ để nhóm dữ liệu theo đội và sau đó tính giá trị trung bình ($mean()$) cho tất cả các cột số trong mỗi nhóm. $reset\_index()$ chuyển cột 'Team' từ index trở lại thành cột dữ liệu thông thường.

```
# 4. Xác định đội có giá trị trung bình cao nhất cho mỗi thống kê
# Đảm bảo chỉ các cột số được bao gồm trong tính toán trung bình theo nhóm
numeric_cols_for_mean = [col for col in numeric_columns if pd.api.types.is_numeric_dtype(df_calc[col])]
if not numeric_cols_for_mean:
    print("Không có cột số nào khả dụng để tính toán trung bình của đội.")
    highest_teams_df = pd.DataFrame() # Tạo DataFrame rỗng
else:
    # Tính trung bình cho từng đội theo các cột số
    team_means = df_calc.groupby("Team")[numeric_cols_for_mean].mean().reset_index()
    highest_teams = []
    for stat in numeric_cols_for_mean:
        # Kiểm tra xem cột có tồn tại và có dữ liệu trước khi tìm giá trị lớn nhất
        if stat in team_means.columns and not team_means[stat].isnull().all():
            # Tìm hàng có giá trị trung bình lớn nhất cho thống kê hiện tại
            max_row = team_means.loc[team_means[stat].idxmax()]
            highest_teams.append({
                "Thống kê": stat,
                "Đội": max_row["Team"],
                "Giá trị Trung bình": round(max_row[stat], 2) })
        else:
            print(f"Bỏ qua tính toán trung bình cao nhất cho '{stat}' do thiếu dữ liệu hoặc tất cả là NaN.")
    # Tạo DataFrame từ kết quả
    highest_teams_df = pd.DataFrame(highest_teams)
# Lưu thống kê đội có giá trị cao nhất vào tệp highest_team_stats.csv trong thư mục 'csv'
highest_team_stats_path = os.path.join(csv_dir, "highest_team_stats.csv")
highest_teams_df.to_csv(highest_team_stats_path, index=False, encoding="utf-8-sig")
print(f"Đã lưu thống kê đội có giá trị cao nhất vào {highest_team_stats_path} với {highest_teams_df.shape[0]} hàng.")
```

- **Tìm Đội có Trung bình Cao nhất**: Lặp qua từng cột thống kê số. Đối với mỗi thống kê, nó tìm chỉ số của hàng có giá trị lớn nhất trong cột đó bằng $idxmax()$ và sử dụng .loc để truy xuất toàn bộ hàng đó. Thông tin về thống kê, tên đội và giá trị trung bình cao nhất được lưu vào danh sách $highest\_teams$.
- **Tạo và Lưu DataFrame**: Danh sách $highest\_teams$ được chuyển đổi thành DataFrame ($highest\_teams\_df$). DataFrame này sau đó được lưu vào tệp $highest\_team\_stats$ trong thư mục csv.

2.7 Xác định Đội có Thành tích Tốt nhất

Phần cuối cùng của script cố gắng xác định "đội có thành tích tốt nhất"dựa trên việc đội đó dẫn đầu trong nhiều thống kê được coi là "tích cực".

```
1: # 5. Xác định đội có thành tích tốt nhất
2: # Định nghĩa các thống kê mà giá trị thấp hơn là tốt hơn (ví dụ: số bàn thua, thẻ phạt,
       mất bóng)
3: negative_stats = [ "GA90", "crdY", "crdR", "Lost", "Mis", "Dis", "Fls", "Off", "Aerl
       Lost"]
4: # Đảm bảo highest_teams_df không rỗng trước khi tiếp tục
5: if not highest_teams_df.empty:
6:     # Lọc ra các thống kê "tích cực"(không nằm trong danh sách negative_stats) thực sự
       có trong DataFrame
7:     # Sử dụng copy()
8:     positive_stats_df = highest_teams_df[~highest_teams_df["Thống
       kê"].isin(negative_stats)].copy()
9:     if not positive_stats_df.empty:
10:         # Đếm số lần mỗi đội đứng đầu trong các thống kê tích cực
11:         team_wins = positive_stats_df["Đội"].value_counts()
12:         if not team_wins.empty:
13:             # Xác định đội có số lần đứng đầu nhiều nhất
14:             best_team = team_wins.idxmax()
15:             win_count = team_wins.max()
16:             print(f"\nĐội có thành tích tốt nhất mùa giải Premier League 2024-2025 (dựa
       trên việc dẫn đầu nhiều thống kê tích cực nhất) là: {best_team}")
17:             print(f"Họ dẫn đầu trong {win_count} trên tổng số {len(positive_stats_df)}
       thống kê tích cực.")
18:         else:
19:             print("\nKhông thể xác định đội có thành tích tốt nhất vì không có đội nào dẫn
       đầu trong các thống kê tích cực.")
20:     else:
21:         print("\nKhông tìm thấy thống kê tích cực nào để xác định đội có thành tích tốt
       nhất.")
22: else:
23:     print("\nKhông thể xác định đội có thành tích tốt nhất vì dữ liệu thống kê đội có giá
       trị cao nhất không khả dụng.")
```

- **Định nghĩa Thống kê Tiêu cực**: Một danh sách *negative_stats* được tạo để liệt kê các thống kê mà giá trị thấp hơn được coi là tốt hơn (ví dụ: số bàn thua trung bình mỗi 90 phút, thẻ phạt).
- **Lọc Thống kê Tích cực**: Lọc DataFrame *highest_teams_df* để chỉ giữ lại các hàng mà thống kê không nằm trong danh sách *negative_stats*. Đây được coi là các thống kê "tích cực"(ví dụ: số bàn thắng, kiến tạo, v.v., nơi giá trị cao hơn là tốt hơn).
- **Đếm Số lần Dẫn đầu**: Sử dụng *value_counts*() trên cột "Đội"của DataFrame thống kê tích cực để đếm số lần mỗi đội xuất hiện (tức là số lần mỗi đội có giá trị trung bình cao nhất trong một thống kê tích cực).
- **Xác định Đội Tốt nhất**: Tìm đội có số lần xuất hiện nhiều nhất trong kết quả của *value_counts*(). Đội này được coi là đội có thành tích tốt nhất dựa trên tiêu chí này. Kết quả được in ra console.

2.8 Kết luận

Script Python này là một công cụ mạnh mẽ để phân tích dữ liệu thống kê bóng đá. Nó thực hiện các bước sau:

- Tải và làm sạch dữ liệu từ tệp CSV.
- Xác định và lưu trữ danh sách top 3 cầu thủ (cao nhất và thấp nhất) cho từng thống kê.
- Tính toán và lưu trữ các thống kê mô tả (trung vị, trung bình, độ lệch chuẩn) ở cấp độ giải đấu và cấp độ đội.
- Tạo và lưu các biểu đồ histogram để trực quan hóa phân phối dữ liệu.
- Xác định đội có giá trị trung bình cao nhất cho mỗi thống kê.
- Cố gắng xác định đội có thành tích tổng thể tốt nhất dựa trên việc dẫn đầu trong các thống kê tích cực.

Kết quả của script được lưu vào các tệp $top\_3.txt, results2.csv, highest\_team\_stats.csv$ và các tệp hình ảnh histogram trong các thư mục được chỉ định. Điều này cho phép người dùng dễ dàng truy cập và diễn giải các phân tích đã thực hiện.

Chương 3

Phân tích và Phân cụm Dữ liệu Cầu thủ Bóng đá

3.1 Giới thiệu

Chương này trình bày chi tiết quá trình phân tích và phân cụm dữ liệu cầu thủ bóng đá sử dụng thuật toán K-Means trong Python. Mục tiêu là nhóm các cầu thủ có đặc điểm thống kê tương đồng lại với nhau, giúp hiểu rõ hơn về các loại hình cầu thủ dựa trên hiệu suất của họ. Dữ liệu đầu vào được lấy từ tệp *result.csv* và kết quả phân tích, bao gồm các biểu đồ trực quan, sẽ được lưu vào thư mục *png*.

3.2 Chuẩn bị Dữ liệu

Quá trình phân tích bắt đầu bằng việc tải và chuẩn bị dữ liệu

3.2.1 Tải Dữ liệu

Đoạn mã thực hiện tải dữ liệu từ tệp *result.csv* nằm trong thư mục con *csv* của thư mục gốc được chỉ định (*base_dir*). Đoạn mã này sử dụng thư viện *pandas* để đọc tệp CSV.

```
# Định nghĩa đường dẫn đầy đủ đến file result.csv trong thư mục csv
result_path = os.path.join(csv_dir, "result.csv")
# Tải dữ liệu
try:
    df = pd.read_csv(result_path, encoding="utf-8-sig")
    print(f"Đã tải dữ liệu thành công từ {result_path}")
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy result.csv tại {result_path}")
    exit() # Thoát nếu không tìm thấy file
except Exception as e:
    print(f"Đã xảy ra lỗi khi tải dữ liệu: {e}")
    exit() # Thoát nếu có lỗi khác
```

Cơ chế xử lý lỗi $try-except$ được thêm vào để bắt các trường hợp tệp không tồn tại hoặc các lỗi khác trong quá trình tải, đảm bảo chương trình không bị dừng đột ngột

3.2.2 Chọn Đặc trưng và Xử lý Giá trị Thiếu

Để thực hiện phân cụm, chỉ các cột chứa dữ liệu số và có ý nghĩa thống kê mới được chọn. Các cột định danh như Tên cầu thủ (*Player*), Quốc gia (*Nation*), Đội (*Team*), Vị trí (*Position*) bị loại bỏ. Sau khi chọn các cột số, đoạn mã kiểm tra xem có bất kỳ cột

```
# Chọn các cột số cho phân cụm (loại trừ các cột không phải số và cột định danh)
numeric_columns = [col for col in df.columns if col not in ["Player", "Nation", "Team",
    "Position"]]
if not numeric_columns:
  print("Lỗi: Không tìm thấy cột số nào để phân cụm sau khi loại trừ các cột định danh.")
  exit()
X = df[numeric_columns]
# Xử lý giá trị thiếu bằng cách điền giá trị trung bình
imputer = SimpleImputer(strategy="mean")
X_imputed = imputer.fit_transform(X)
```

số nào được chọn hay không. Nếu không có, chương trình sẽ thoát. Tiếp theo, sử dụng *SimpleImputer* từ *sklearn.impute* với chiến lược điền giá trị trung bình (*strategy* = "*mean*") để xử lý các giá trị thiếu (NaN) trong tập dữ liệu số đã chọn. Điều này là cần thiết vì hầu hết các thuật toán phân cụm không thể xử lý dữ liệu có giá trị thiếu.

3.2.3 Chuẩn hóa Dữ liệu

Phân cụm K-Means nhạy cảm với thang đo của dữ liệu. Do đó, dữ liệu cần được chuẩn hóa để tất cả các đặc trưng có cùng thang đo, ngăn chặn các đặc trưng có giá trị lớn hơn chi phối quá trình phân cụm.

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
"'StandardScaler' từ 'sklearn.preprocessing' được sử dụng để chuẩn hóa dữ liệu bằng cách loại bỏ giá trị trung bình và chia cho độ lệch chuẩn, kết quả là dữ liệu có giá trị trung bình bằng 0 và phương sai bằng 1.
## 3. Xác định Số lượng Cụm Tối ưu
Việc chọn số lượng cụm (K) là một bước quan trọng trong K-Means. Đoạn mã này sử dụng phương pháp Elbow để hỗ trợ việc lựa chọn này.
"'python
# Xác định số lượng cụm tối ưu bằng phương pháp Elbow
wcss = []
max_clusters = 10
# Đảm bảo có đủ mẫu cho ít nhất 2 cụm và max_clusters không lớn hơn số lượng mẫu
if X_scaled.shape[0] < 2:
    print("Lỗi: Không đủ điểm dữ liệu để thực hiện phân cụm.")
    exit()
max_clusters = min(max_clusters, X_scaled.shape[0])
if max_clusters < 2:
    print("Lỗi: Không thể xác định số cụm tối ưu với ít hơn 2 điểm dữ liệu.")
    exit()
for i in range(2, max_clusters + 1):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
# Vẽ biểu đồ Elbow
optimal_k = 4 # Giả định ban đầu, cần kiểm tra biểu đồ
plt.figure(figsize=(8, 5))
plt.plot(range(2, max_clusters + 1), wcss, marker='o', linestyle='-', color='b')
plt.axvline(x=optimal_k, color='r', linestyle='--', label=f'Số cụm tối ưu = optimal_k')
plt.title('Phương pháp Elbow để xác định số lượng cụm tối ưu')
plt.xlabel('Số lượng cụm (K)')
plt.ylabel('WCSS (Tổng bình phương trong cụm)')
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
# Lưu biểu đồ elbow vào thư mục png
elbow_plot_path = os.path.join(png_dir, "elbow_plot.png")
plt.savefig(elbow_plot_path, format='png', dpi=300, bbox_inches='tight')
plt.close()
print(f"Biểu đồ elbow đã được lưu vào elbow_plot_path")
```

Phương pháp Elbow tính toán Tổng bình phương trong cụm (WCSS - Within-Cluster Sum of Squares) cho các số lượng cụm khác nhau. WCSS là tổng khoảng cách bình phương giữa mỗi điểm dữ liệu và tâm cụm của nó. Khi số lượng cụm tăng lên, WCSS thường giảm. Điểm "khuỷu tay"trên biểu đồ WCSS so với số lượng cụm (nơi tốc độ giảm của WCSS chậm lại đáng kể) được coi là số lượng cụm tối ưu. Đoạn mã vẽ biểu đồ này và lưu lại dưới dạng tệp PNG. Một giá trị *optimal_k* ban đầu được đặt là 4, nhưng người phân tích cần xem xét biểu đồ được tạo ra để xác nhận hoặc điều chỉnh giá trị này.

3.3 Áp dụng K-Means và Gán Nhãn Cụm

Sau khi xác định được số lượng cụm tối ưu, thuật toán K-Means được áp dụng cho dữ liệu đã chuẩn hóa. Đoạn mã kiểm tra tính hợp lệ của *optimal_k* trước khi áp dụng K-Means.

```
# Áp dụng K-means với số lượng cụm tối ưu
if optimal_k < 2 or optimal_k > X_scaled.shape[0]:
    print(f"Lỗi: Giá trị optimal_k không hợp lệ ({optimal_k}). Vui lòng chọn giá trị từ 2 đến {X_scaled.shape[0]}.")
    exit()
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X_scaled)
# Thêm nhãn cụm vào dataframe gốc
df['Cluster'] = cluster_labels
```

Thuật toán K-Means được khởi tạo với số lượng cụm đã chọn và $random\_state = 42$ để đảm bảo kết quả có thể tái lập. $n\_init = 10$ chỉ định rằng thuật toán sẽ được chạy 10 lần với các điểm khởi tạo tâm cụm khác nhau và kết quả tốt nhất sẽ được chọn. Kết quả của quá trình phân cụm là một mảng chứa nhãn cụm cho mỗi điểm dữ liệu, sau đó được thêm vào DataFrame gốc dưới một cột mới có tên *Cluster*.

3.4 Trực quan hóa Kết quả Phân cụm bằng PCA

Để trực quan hóa các cụm trong không gian 2 chiều, kỹ thuật Phân tích Thành phần Chính (PCA) được sử dụng để giảm chiều dữ liệu.

```
# PCA để trực quan hóa 2D
if X_scaled.shape[0] < 2:
  print("Lỗi: Không đủ điểm dữ liệu cho PCA.")
  exit()
n_components_pca = min(2, X_scaled.shape[1])
pca = PCA(n_components=n_components_pca)
X_pca = pca.fit_transform(X_scaled)
# Phương sai giải thích cho nhãn trục
explained_variance = pca.explained_variance_ratio_ total_variance = sum(explained_variance)
# Vẽ biểu đồ các cụm 2D với chú thích chi tiết
plt.figure(figsize=(12, 8))
colors = plt.cm.viridis(np.linspace(0, 1, optimal_k))
handles = []
for cluster in range(optimal_k):
  mask = cluster_labels == cluster
  if np.any(mask):
    scatter = plt.scatter( X_pca[mask, 0], X_pca[mask, 1], s=100, alpha=0.7, color=colors[cluster], label=f'Cụm {cluster}' )
    handles.append(scatter)
  else:
    print(f"Cảnh báo: Cụm cluster trống và sẽ không được vẽ.")
plt.legend(handles=handles, title='Các cụm', loc='best', fontsize=10)
# Chú thích 5 cầu thủ ghi bàn hàng đầu
if 'Gls' in df.columns:
  num_players_to_annotate = min(5, len(df))
  top_players = df.nlargest(num_players_to_annotate, 'Gls').index
  for idx in top_players:
    plt.annotate( df.loc[idx, 'Player'], (X_pca[idx, 0], X_pca[idx, 1]), fontsize=8, xytext=(5, 5), textcoords='offset points', bbox=dict(boxstyle="round,pad=0.3", edgecolor="black", facecolor="white", alpha=0.8) )
else:
  print("Cảnh báo: Không tìm thấy cột 'Gls'. Không thể chú thích các cầu thủ hàng đầu theo bàn thắng.")
plt.title(f'Trực quan hóa PCA các cụm cầu thủ (K={optimal_k}, {total_variance:.1%} Phương sai được giải thích)', fontsize=12)
plt.xlabel(f'Thành phần PCA 1 ({explained_variance[0]:.1%} Phương sai)', fontsize=10)
plt.ylabel(f'Thành phần PCA 2 ({explained_variance[1]:.1%} Phương sai)', fontsize=10)
plt.grid(True, linestyle='--', alpha=0.7)
# Lưu biểu đồ cụm PCA vào thư mục png
pca_plot_path = os.path.join(png_dir, "pca_cluster_plot.png")
plt.savefig(pca_plot_path, format='png', dpi=300, bbox_inches='tight')
plt.close()
print(f"Biểu đồ cụm PCA đã được lưu vào {pca_plot_path}")
```

PCA được áp dụng để giảm số chiều của dữ liệu đã chuẩn hóa xuống còn 2 thành phần chính. Biểu đồ phân tán sau đó được vẽ, với mỗi điểm đại diện cho một cầu thủ và được tô màu theo cụm mà nó thuộc về. Biểu đồ bao gồm chú giải cho từng cụm và chú thích tên của 5 cầu thủ ghi bàn hàng đầu (dựa trên cột 'Gls') để cung cấp thêm ngữ cảnh. Tiêu đề biểu đồ và nhãn trục hiển thị thông tin về số lượng cụm và tỷ lệ phương sai được giải thích bởi các thành phần PCA. Biểu đồ này giúp trực quan hóa sự phân bố của các cụm trong không gian 2 chiều.

3.5 Phân tích Kết quả Phân cụm

Sau khi phân cụm, việc phân tích các đặc điểm của từng cụm là cần thiết để hiểu ý nghĩa của chúng. Đoạn mã in ra kích thước của mỗi cụm (số lượng cầu thủ trong mỗi cụm) bằng

```
# Phân tích các cụm
print("\nKích thước các cụm:")
print(df['Cluster'].value_counts())
# Thống kê tóm tắt cụm
summary_cols = ['Gls', 'Ast', 'Tkl', 'Save%']
available_summary_cols = [col for col in summary_cols if col in df.columns]
if available_summary_cols:
    cluster_summary = df.groupby('Cluster')[available_summary_cols].mean().round(2)
    print("\nTóm tắt cụm (Giá trị trung bình):")
    print(cluster_summary)
else:
    print("\nCảnh báo: Không tìm thấy cột nào trong số các cột tóm tắt được yêu cầu (Gls, Ast, Tkl, Save%) trong dataframe.")
# Phương sai được giải thích bởi các thành phần PCA
print(f"\nTỷ lệ phương sai được giải thích bởi PCA: {explained_variance}")
print(f"Tổng phương sai được giải thích: {total_variance:.2%}")
```

cách sử dụng *value_counts*() trên cột *Cluster*. Tiếp theo, nó tính toán giá trị trung bình của một số cột thống kê quan trọng (Bàn thắng - 'Gls', Kiến tạo - 'Ast', Tắc bóng - 'Tkl', Tỷ lệ cứu thua - 'Save%') cho mỗi cụm. Điều này giúp xác định các đặc điểm nổi bật của từng nhóm cầu thủ. Ví dụ, một cụm có giá trị trung bình 'Gls' cao có thể đại diện cho các tiền đạo, trong khi một cụm có 'Save%' cao có thể là các thủ môn. Cuối cùng, tỷ lệ phương sai được giải thích bởi các thành phần PCA cũng được in ra để đánh giá mức độ hiệu quả của việc giảm chiều dữ liệu.

3.6 Nhận xét về Số lượng Nhóm Người chơi và Kết quả Phân cụm

Dựa trên phân tích đã thực hiện, việc phân loại người chơi vào các nhóm là một cách hiệu quả để hiểu rõ hơn về các loại hình cầu thủ dựa trên dữ liệu thống kê. Số lượng nhóm tối ưu được gợi ý bởi phương pháp Elbow. Biểu đồ Elbow (được lưu dưới dạng *elbow_plot.png*) hiển thị WCSS (Tổng bình phương trong cụm) giảm dần khi số lượng cụm tăng. Điểm "khuỷu tay"trên biểu đồ, nơi tốc độ giảm của WCSS chậm lại đáng kể, thường được coi là số lượng cụm tối ưu. Trong đoạn mã này, giá trị *optimal_k* ban đầu

được đặt là 4. Việc kiểm tra trực quan biểu đồ Elbow là rất quan trọng để xác nhận xem K=4 có thực sự là điểm "khuỷu tay"rõ ràng hay không, hoặc liệu một số lượng cụm khác (ví dụ: 3 hoặc 5) có thể phù hợp hơn.
Việc phân loại cầu thủ thành K nhóm (với K là số cụm tối ưu đã chọn) giúp chúng ta phân khúc các cầu thủ dựa trên các chỉ số hiệu suất của họ. Mỗi cụm có khả năng đại diện cho một "kiểu"cầu thủ nhất định, ví dụ: tiền đạo, tiền vệ phòng ngự, thủ môn, v.v., mặc dù việc gán nhãn cụ thể cho từng cụm cần dựa vào việc phân tích chi tiết các giá trị trung bình của các đặc trưng trong từng cụm (như đã hiển thị trong "Tóm tắt cụm").

Về kết quả phân cụm, biểu đồ PCA trực quan hóa sự phân tách của các cụm trong không gian 2 chiều. Nếu các cụm được phân tách rõ ràng trên biểu đồ PCA, điều này cho thấy thuật toán K-Means đã tìm thấy các nhóm cầu thủ tương đối khác biệt dựa trên các đặc trưng đã sử dụng. Tỷ lệ phương sai được giải thích bởi các thành phần PCA (được in ra ở cuối quá trình) cho biết mức độ mà hai thành phần chính này đại diện cho sự biến thiên tổng thể của dữ liệu gốc. Tỷ lệ này càng cao thì biểu đồ PCA càng phản ánh chính xác cấu trúc phân cụm trong không gian nhiều chiều ban đầu.

Nhìn chung, kết quả phân cụm cung cấp một cái nhìn có cấu trúc về tập dữ liệu cầu thủ, cho phép xác định các nhóm cầu thủ có hồ sơ hiệu suất tương tự. Điều này có thể hữu ích cho nhiều mục đích như tìm kiếm cầu thủ có đặc điểm phù hợp với một vai trò nhất định, phân tích điểm mạnh/điểm yếu của các nhóm cầu thủ khác nhau, hoặc thậm chí là cơ sở cho các mô hình dự đoán hiệu suất trong tương lai. Tuy nhiên, cần lưu ý rằng chất lượng của việc phân cụm phụ thuộc nhiều vào chất lượng và sự phù hợp của các đặc trưng đầu vào.

Chương 4

Phân Tích Dữ liệu Cầu thủ, Thu thập Giá trị Chuyển nhượng và Dự đoán Giá trị bằng Machine Learning tại Premier League

4.1 Giới thiệu

Báo cáo này mô tả một quy trình xử lý dữ liệu đa giai đoạn nhằm phân tích thông tin cầu thủ từ tệp dữ liệu ban đầu (*result.csv*), thu thập dữ liệu bổ sung từ web về giá trị chuyển nhượng đã xác nhận và giá trị chuyển nhượng ước tính (ETV), sau đó áp dụng mô hình Machine Learning (Hồi quy Tuyến tính) để dự đoán giá trị chuyển nhượng dựa trên các chỉ số thống kê và thuộc tính của cầu thủ. Quy trình này sử dụng các thư viện Python mạnh mẽ như *pandas* để xử lý dữ liệu, selenium để tự động hóa trình duyệt và thu thập dữ liệu web, *fuzzywuzzy* để so khớp tên cầu thủ, và *sklearn* để xây dựng và huấn luyện mô hình Machine Learning.

4.2 Mục tiêu

Các mục tiêu chính của toàn bộ quy trình bao gồm:

- Lọc và xử lý dữ liệu cầu thủ ban đầu dựa trên thời gian thi đấu.
- Thu thập thông tin giá trị chuyển nhượng đã xác nhận cho các cầu thủ từ một nguồn web.
- Thu thập dữ liệu giá trị chuyển nhượng ước tính (ETV) chi tiết cho nhiều cầu thủ từ cùng nguồn web.
- Sử dụng kỹ thuật so khớp chuỗi mờ (fuzzy matching) để liên kết dữ liệu từ các nguồn khác nhau.
- Xây dựng các mô hình Machine Learning (Linear Regression) riêng cho từng vị trí cầu thủ (Thủ môn, Hậu vệ, Tiền vệ, Tiền đạo) dựa trên các chỉ số thống kê phù hợp.
- Dự đoán giá trị chuyển nhượng cho các cầu thủ dựa trên mô hình đã huấn luyện.

- Tổng hợp và lưu trữ dữ liệu gốc, dữ liệu thu thập và kết quả dự đoán vào các tệp CSV có cấu trúc.

4.3 Phương pháp thực hiện

Quy trình được thực hiện theo ba giai đoạn liên tiếp:

4.3.1 Giai đoạn 1: Lọc cầu thủ theo phút thi đấu và thu thập giá chuyển nhượng đã xác nhận

Mục đích: Xác định các cầu thủ chủ chốt dựa trên thời gian thi đấu và thu thập thông tin về các vụ chuyển nhượng thực tế của họ.
Nguồn dữ liệu: Tệp $result.csv$ và các trang danh sách chuyển nhượng đã xác nhận trên $footballtransfers.com$.
Các bước chính:

- Tải $result.csv$ và lọc ra các cầu thủ có $Minutes > 900$. Kết quả được lưu vào $players\_over\_900\_minutes.csv$.
- Sử dụng Selenium ở chế độ headless để crawl dữ liệu tên cầu thủ và giá chuyển nhượng đã xác nhận từ các trang chuyển nhượng.
- Áp dụng hàm $shorten\_name$ (lấy 2 từ đầu) cho tên cầu thủ từ cả hai nguồn.
- Thực hiện so khớp mờ ($fuzzywuzz$ với $fuzz.token\_sort\_ratio$, ngưỡng 85) giữa tên cầu thủ crawl được và tên cầu thủ trong $players\_over\_900\_minutes.csv$.
- Lưu các cặp (Tên cầu thủ, Giá trị chuyển nhượng đã xác nhận) đã khớp vào $player\_transfer\_fee.csv$.

4.3.2 Giai đoạn 2: Thu thập giá trị chuyển nhượng ước tính (ETV) theo vị trí

Mục đích: Thu thập dữ liệu ETV chi tiết để sử dụng làm biến mục tiêu cho mô hình dự đoán sau này.
Nguồn dữ liệu: Tệp $result.csv$ và các trang danh sách cầu thủ cùng ETV trên $footballtransfers.com$
Các bước chính:

- Tải $result.csv$.
- Áp dụng hàm $shorten\_name$ (phiên bản nâng cao xử lý các trường hợp đặc biệt) cho tên cầu thủ từ $result.csv$ và tạo từ điển tra cứu Vị trí, Tên gốc.
- Sử dụng Selenium ở chế độ headless để crawl dữ liệu tên cầu thủ, vị trí và ETV từ các trang danh sách cầu thủ trên $footballtransfers.com$.
- Áp dụng hàm $shorten\_name$ (phiên bản nâng cao) cho tên cầu thủ crawl được.
- Thực hiện so khớp mờ ($fuzzywuzzy$ với $fuzz.token\_sort\_ratio$, ngưỡng 80) giữa tên cầu thủ crawl được và tên cầu thủ trong $result.csv$.

- Với mỗi cầu thủ khớp, trích xuất Vị trí và Tên gốc từ từ điển tra cứu và phân tích giá trị ETV ($parse\_etv$ để chuyển đổi chuỗi như '€20M' thành số).
- Lưu trữ dữ liệu đã khớp (Tên gốc, Vị trí, ETV) vào các danh sách riêng theo vị trí (GK, DF, MF, FW), giữ nguyên thứ tự nhóm.
- Kết hợp các danh sách này và lưu vào $all\_estimate\_transfer\_fee.csv$.

4.3.3 Giai đoạn 3: Dự đoán Giá trị Chuyển nhượng Ước tính bằng Mô hình Hồi quy Tuyến tính

Mục đích: Xây dựng mô hình Machine Learning sử dụng các chỉ số thống kê cầu thủ để dự đoán ETV.
Nguồn dữ liệu: Tệp $result.csv$ (chứa các chỉ số thống kê, Team, Nation, Age, Minutes, Position) và $all\_estimate\_transfe$

- Lọc cầu thủ từ $result.csv$ theo vị trí chính.

  Thực hiện so khớp mờ ($fuzzywuzzy$ với $fuzz.token\_sort\_ratio$, ngưỡng 90) giữa tên cầu thủ trong $result.csv$ (đã lọc theo vị trí) và danh sách tên cầu thủ từ $all\_estimate\_transfer\_fee.csv$. Mục đích là liên kết các chỉ số thống kê từ $result.csv$ với ETV từ $all\_estimate\_transfer\_fee.csv$.

  Lấy giá trị ETV tương ứng cho các cầu thủ đã khớp.

  Xử lý các cầu thủ không khớp (ghi nhận lại danh sách).

  Chuẩn bị dữ liệu cho mô hình:
  - Chọn các đặc trưng (features) đã được định nghĩa cho từng vị trí trong $positions\_config$.
  - Xử lý giá trị thiếu (điền median/0 cho số, 'Unknown' cho phân loại).
  - Áp dụng phép biến đổi $np.log1p$ cho các đặc trưng số.
  - Áp dụng trọng số cho các đặc trưng quan trọng (như $important\_features$, $Minutes$, Age).

  Xây dựng và huấn luyện mô hình:
  - Tạo tập dữ liệu huấn luyện (df_ml_train) bao gồm các cầu thủ đã khớp và có giá trị ETV hợp lệ.
  - Sử dụng $ColumnTransformer$ để tiền xử lý dữ liệu (StandardScaler cho số, OneHotEncoder cho phân loại).
  - Xây dựng $Pipeline$ kết hợp tiền xử lý và mô hình $LinearRegression$.
  - Huấn luyện Pipeline trên tập dữ liệu huấn luyện (hoặc chia nhỏ nếu đủ dữ liệu).

  Dự đoán giá trị: Sử dụng mô hình đã huấn luyện để dự đoán giá trị chuyển nhượng cho tất cả các cầu thủ đã so khớp thành công ở vị trí hiện tại (bao gồm cả những người không có ETV gốc).

  Xử lý kết quả dự đoán: Cắt giá trị dự đoán trong khoảng hợp lý, định dạng giá trị dự đoán và ETV gốc sang đơn vị triệu (làm tròn 2 chữ số).

  Chọn các cột đầu ra chuẩn (Player, Team, Nation, Position, Actual_Transfer_Value_M, Predicted_Transfer_Value_M).

  Kết hợp kết quả dự đoán từ tất cả các vị trí vào một DataFrame duy nhất.

  Sắp xếp kết quả theo giá trị dự đoán giảm dần.

  Lưu DataFrame cuối cùng vào tệp $ml\_transfer\_values\_linear.csv$.

  In danh sách các cầu thủ từ $result.csv$ không tìm thấy so khớp trong dữ liệu ETV.

4.4 Kết quả

Quy trình đã tạo ra ba tệp dữ liệu trong thư mục *csv*:

- `players_over_900_minutes.csv`: Danh sách cầu thủ có hơn 900 phút thi đấu.
- `playe_transfer_fee.csv`: Thông tin giá trị chuyển nhượng đã xác nhận cho các cầu thủ được so khớp từ danh sách trên.
- `all_estimate_transfer_fee.csv`: Thông tin ETV chi tiết cho các cầu thủ Premier League, được nhóm theo vị trí (GK, DF, MF, FW).
- `ml_transfer_values_linear.csv`: Tệp kết quả cuối cùng chứa thông tin Player, Team, Nation, Position, ETV thực tế (nếu có) và Giá trị chuyển nhượng được dự đoán bởi mô hình Linear Regression, tất cả đều được làm tròn và trình bày ở đơn vị triệu ($).

4.5 Bàn luận

Quy trình này cho thấy một cách tiếp cận toàn diện để làm giàu dữ liệu cầu thủ bằng cách kết hợp thông tin từ nhiều nguồn (file CSV cục bộ và web scraping).
Việc sử dụng Fuzzy Matching là rất quan trọng để kết nối dữ liệu từ các nguồn có thể có sự khác biệt về định dạng tên. Các hàm *shorten_name* và ngưỡng so khớp đóng vai trò quyết định trong độ chính xác.
Giai đoạn áp dụng Machine Learning cho phép ước tính giá trị chuyển nhượng cho các cầu thủ dựa trên hiệu suất và thuộc tính của họ, sử dụng ETV thu thập được làm cơ sở huấn luyện.
Việc xây dựng mô hình riêng cho từng vị trí (GK, DF, MF, FW) và lựa chọn các đặc trưng thống kê phù hợp cho từng vị trí là một điểm mạnh, phản ánh sự khác biệt về vai trò và chỉ số quan trọng của từng nhóm cầu thủ.
Các bước tiền xử lý dữ liệu trong mô hình (xử lý giá trị thiếu, biến đổi $log1p$, tăng trọng số đặc trưng) là cần thiết để cải thiện hiệu suất và độ tin cậy của mô hình Linear Regression.
Mô hình Linear Regression cung cấp một ước tính tuyến tính về giá trị chuyển nhượng. Đối với một thị trường phức tạp như chuyển nhượng cầu thủ, các mô hình nâng cao hơn có thể cần thiết để nắm bắt hết các yếu tố phi tuyến tính. Tuy nhiên, đây là một điểm khởi đầu hợp lý.
Danh sách cầu thủ không được khớp trong Giai đoạn 3 chỉ ra những trường hợp mà quá trình so khớp cần được xem xét lại hoặc dữ liệu ETV không đầy đủ cho các cầu thủ đó.

4.6 Kết luận

Ba tập lệnh đã phối hợp thành công để thực hiện một quy trình từ thu thập dữ liệu thô đến dự đoán giá trị bằng Machine Learning. Kết quả là tệp $ml\_transfer\_values\_linear.csv$ cung cấp một cái nhìn tổng hợp về các cầu thủ đã được xử lý, bao gồm cả giá trị chuyển nhượng ước tính thực tế (nếu có) và giá trị dự đoán dựa trên mô hình. Dữ liệu này là một tài nguyên quý giá cho các phân tích chuyên sâu hơn về giá trị cầu thủ, hiệu suất và xu hướng thị trường chuyển nhượng tại Premier League.

TỔNG KẾT

Bài tập lớn này đã hoàn thành các mục tiêu đề ra, chứng minh khả năng ứng dụng Python trong phân tích dữ liệu bóng đá chuyên sâu. Từ việc thu thập dữ liệu thô từ các nguồn web, chuẩn hóa và làm sạch dữ liệu, đến việc thực hiện các phân tích thống kê mô tả, xác định nhóm cầu thủ thông qua phân cụm K-Means, và cuối cùng là xây dựng mô hình Hồi quy Tuyến tính để dự đoán giá trị chuyển nhượng, mỗi giai đoạn đều cho thấy sức mạnh và tính linh hoạt của Python cùng các thư viện hỗ trợ.

Các kết quả phân tích thống kê đã cung cấp cái nhìn định lượng về hiệu suất cá nhân của cầu thủ và hiệu suất tổng thể của các đội. Quá trình phân cụm giúp nhận diện các nhóm cầu thủ có đặc điểm tương đồng, hỗ trợ việc phân loại và đánh giá tiềm năng. Đặc biệt, mô hình dự đoán giá trị chuyển nhượng đã mở ra khả năng ước lượng giá trị thị trường của cầu thủ dựa trên các chỉ số hiệu suất, một ứng dụng quan trọng trong ngành công nghiệp bóng đá hiện đại.

Tuy nhiên, báo cáo cũng nhận thấy một số khía cạnh có thể cải thiện, như tính ổn định của quá trình thu thập dữ liệu web trước sự thay đổi cấu trúc trang nguồn, và khả năng mở rộng của mô hình dự đoán với các yếu tố phi tuyến tính hoặc dữ liệu lịch sử phong phú hơn. Dù vậy, những kết quả đạt được đã khẳng định hiệu quả của việc sử dụng Python làm công cụ chính cho các bài toán phân tích dữ liệu phức tạp trong lĩnh vực thể thao.

Tóm lại, bài tập lớn này không chỉ củng cố kỹ năng lập trình Python mà còn trang bị kiến thức và kinh nghiệm quý báu trong lĩnh vực Khoa học Dữ liệu và Machine Learning, đặc biệt trong bối cảnh dữ liệu thể thao đang ngày càng phổ biến và có giá trị.