HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN CƠ BẢN 1

BÀI TẬP LỚN

NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn : **Kim Ngọc Bách**

Họ và tên sinh viên : **Vũ Quốc Khải**

Mã sinh viên : **B23DCCE048**

Lớp : **D23CQCE06-B**

Nhóm : **06**

Hà Nội – 2025

LỜI NÓI ĐẦU

Bóng đá, môn thể thao vua, từ lâu đã vượt khỏi ranh giới của những sân cỏ để trở thành một hiện tượng văn hóa toàn cầu, thu hút hàng tỷ người hâm mộ trên khắp thế giới. Trong kỷ nguyên số hóa, bóng đá không chỉ là cuộc chiến trên sân mà còn là cuộc đua dữ liệu ngoài hậu trường. Mỗi đường chuyền, mỗi pha tắc bóng, và mỗi phút thi đấu đều được ghi lại, tạo nên một kho dữ liệu khổng lồ, chứa đựng tiềm năng để mở khóa những hiểu biết sâu sắc về hiệu suất cầu thủ, chiến thuật đội bóng, và giá trị thị trường. Trong bối cảnh đó, trí tuệ nhân tạo (AI) đã nổi lên như một công cụ mạnh mẽ, giúp biến dữ liệu thô thành những quyết định chiến lược. Từ việc phân tích hiệu suất cầu thủ theo thời gian thực, dự đoán giá trị chuyển nhượng, đến tối ưu hóa chiến thuật đội hình, AI đang định hình lại cách các câu lạc bộ, huấn luyện viên, và nhà phân tích tiếp cận bóng đá. Các thuật toán học máy, như K-means để phân cụm cầu thủ hay hồi quy tuyến tính để dự đoán giá trị, không chỉ giúp tiết kiệm thời gian mà còn mang lại độ chính xác và khách quan vượt trội so với phương pháp truyền thống. Báo cáo này trình bày một dự án phân tích toàn diện, ứng dụng AI để xử lý và khai thác dữ liệu cầu thủ trong mùa giải Premier League 2024-2025. Báo cáo tập trung vào bốn nhiệm vụ chính:

1. Thu thập và xử lý dữ liệu: Thu thập thống kê chi tiết của các cầu thủ có thời gian thi đấu trên 90 phút từ FBref.com, bao gồm các tiêu chí
2. Phân tích thống kê: Xác định top 3 cầu thủ có chỉ số cao nhất và thấp, Vẽ biểu đồ histogram để trực quan hóa phân phối chỉ số và xác định đội bóng xuất sắc nhất dựa trên các chỉ số thống kê.
3. Phân cụm cầu thủ: Sử dụng thuật toán K-means để phân loại cầu thủ thành các nhóm dựa trên thống kê, giảm chiều dữ liệu bằng PCA để trực quan hóa các cụm trên biểu đồ 2D.
4. Dự đoán giá trị chuyển nhượng: Thu thập giá trị chuyển nhượng của các cầu thủ có thời gian thi đấu trên 900 phút từ FootballTransfers.com, đề xuất phương pháp ước lượng giá trị sử dụng mô hình học máy, và giải thích cách chọn đặc trưng và mô hình phù hợp.

Báo cáo không chỉ trình bày các giải pháp kỹ thuật mà còn phân tích ý nghĩa của kết quả, từ việc nhận diện các cầu thủ xuất sắc, đánh giá hiệu suất đội bóng, đến khám phá các nhóm cầu thủ có đặc điểm tương đồng. Thông qua dự án này, chúng tôi mong muốn làm sáng tỏ tiềm năng của AI trong việc nâng tầm phân tích bóng đá, đồng thời cung cấp một góc nhìn thực tiễn về cách dữ liệu và công nghệ có thể hỗ trợ các quyết định chiến lược trong môn thể thao vua.

MỤC LỤC

I. Phân tích đề bài

1. Vấn đề 1

- Viết chương trình Python thu thập dữ liệu thống kê cầu thủ thi đấu tại Premier League 2024–2025 từ trang https://fbref.com/en/.
- Chỉ lấy dữ liệu cầu thủ chơi trên 90 phút và lưu dữ liệu vào file result.csv.
- Mỗi cột là một thống kê cụ thể.
- Sắp xếp cầu thủ theo tên đầu tiên.
- Thống kê cầu thủ không có dữ liệu thì ghi là "N/a".
- Xử lý đủ 76 tiêu chí.

2. Vấn đề 2

- Phân tích thống kê các chỉ số đã thu thập và xác định các cầu thủ và đội bóng nổi bật trong mùa giải Premier League 2024–2025.
- Đưa ra 3 cầu thủ cao nhất và thấp nhất của mỗi thống kê và lưu vào file top3.txt.
- Thống kê mô tả cho từng chỉ số (Median, Mean, Std,...).
- Áp dụng cho toàn bộ cầu thủ của từng đội và lưu vào file results2.csv.
- Vẽ biểu đồ Histogram cho từng chỉ số cho toàn bộ đội bóng và giải đấu.
- Xác định đội có giá trị cao nhất theo từng chỉ số.
- Dựa trên phân tích, chỉ ra đội bóng đang thể hiện tốt nhất mùa giải.

3. Vấn đề 3

- Áp dụng thuật toán K-means để phân nhóm cầu thủ dựa trên các chỉ số thống kê.
- Tìm số lượng nhóm tối ưu.
- Giải thích lý do chọn số nhóm đó.
- Bình luận kết quả phân cụm.
- Dùng PCA để giảm số chiều dữ liệu xuống 2 chiều.
- Vẽ biểu đồ 2D hiển thị các nhóm cầu thủ sau phân cụm.

4. Vấn đề 4

- Thu thập dữ liệu giá trị chuyển nhượng của cầu thủ mùa giải 2024–2025 từ trang https://www.footballtransfers.com.
- Chỉ lấy cầu thủ thi đấu trên 900 phút.
- Chọn mô hình học máy phù hợp để dự đoán giá trị cầu thủ.
- Trình bày cách chọn đặc trưng (features) từ các chỉ số thống kê cầu thủ.
- Lý giải tại sao chọn mô hình đó.

II. Thuật toán và phân tích thuật toán

1. Vấn đề 1 & 2

1.1 Phương pháp giải bài

- Thuật toán được triển khai và tiếp cận bằng cách kết hợp web scraping, xử lý dữ liệu, phân tích thống kê, và trực quan hóa để tạo ra một hệ thống phân tích dữ liệu bóng đá tự động. Các phương pháp chính bao gồm:
    + Selenium + BeautifulSoup cho web scraping.
    + Pandas cho xử lý và gộp dữ liệu.
    + Matplotlib cho trực quan hóa.
    + Quản lý file với os và lưu trữ kết quả dưới dạng CSV, TXT, PNG.
    + Xử lý lỗi và tối ưu hóa để đảm bảo tính mạnh mẽ.
- **Lý do vấn đề 1 và vấn đề 2 sử dụng chung thuật toán:** Cách giải và kết quả của vấn đề 2 được triển khai dựa trên thuật toán và ý tưởng của vấn đề 1, khi đã có đáp án kết quả của vấn đề 1 nghĩa là chúng ta đã có cơ sở để xây dựng ý tưởng cho vấn đề 2, vậy nên hoàn toàn có thể dựa vào đáp án của vấn đề 1 để triển khai và đưa ra kết quả của vấn đề 2 nhờ những tiêu chí mà đã được lưu vào file result.csv của vấn đề 1.

1.2 Thuật toán triển khai tổng quát

1.2.1 Các bước hoạt động

Bước 1: Cài đặt và cấu hình

- Import các thư viện cần thiết: selenium, pandas, numpy, re, matplotlib, seaborn.
- Thiết lập Chrome WebDriver chạy ở chế độ không hiển thị (headless).

Bước 2: Truy cập website fbref

- Dùng Selenium để truy cập trang thống kê cầu thủ Premier League từ FBref.

Bước 3: Phân tích html và phân tách dữ liệu

- Tìm các bảng thống kê quan trọng theo ID như:
- "stats_standard" – Thống kê chung
- "stats_keeper" – Thủ môn
- "stats_shooting", "stats_passing", "stats_defense" – Ghi bàn, chuyền bóng, phòng ngự, v.v.

- Một số bảng ẩn trong HTML dạng comment (<!-- -->) nên cần xử lý bằng BeautifulSoup để tìm và parse lại các phần này.

Bước 4: Tiền xử lý dữ liệu

- Đổi tên các cột thành dạng chuẩn hóa (ví dụ: "Player" → "player")
- Chuyển cột "age" từ dạng "23-45" thành số thập phân.
- Chuẩn hóa tên cầu thủ để tránh lỗi khi merge (xóa dấu \\ và các ký tự phụ).

Bước 5: Gộp tất cả các bảng

- Gộp (merge) tất cả các bảng con theo cột "player" và "team".
- Kết quả: Một DataFrame duy nhất chứa đầy đủ thống kê cho mỗi cầu thủ.

Bước 6: Loại bỏ tên cầu thủ trùng và chuyển đổi kiểu dữ liệu

- Loại bỏ các dòng trùng tên cầu thủ (nếu có nhiều dòng thống kê).
- Ép kiểu các cột số sang dạng float hoặc int tùy dữ liệu.

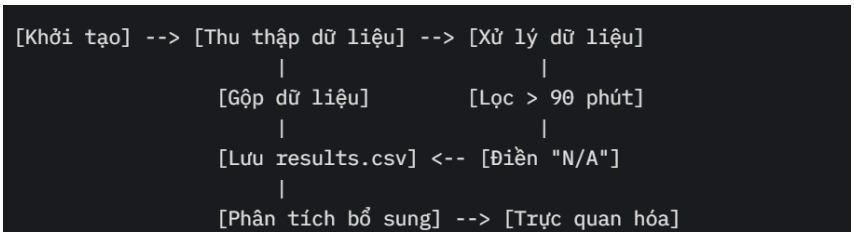
Bước 7: Phân tích dữ liệu

- Tính toán Top 3 cầu thủ theo mỗi chỉ số (ví dụ: "goals", "assists", "passes_completed",...).
- Thống kê tổng hợp: trung bình (mean), trung vị (median), độ lệch chuẩn (std) cho các chỉ số.

Bước 8: Vẽ biểu đồ

- Vẽ biểu đồ histogram cho một số chỉ số (goals, assists, v.v.).

1.2.2 Sơ đồ tổng quát



1.3 Giải thích thuật toán

1.3.1 Giải thích cách thuật toán hoạt động giải quyết vấn đề 1

Giai đoạn 1: Thu thập dữ liệu (Web Scraping)

Mục đích: Trích xuất dữ liệu thống kê từ các bảng trên FBref.com.

Cách thực hiện:

1. **Khởi tạo trình duyệt**

- Sử dụng Selenium với Chrome ở chế độ headless để tải các trang động (JavaScript).
- Cấu hình tùy chọn (--disable-gpu, --no-sandbox) để tối ưu hiệu suất.
- Tự động cài đặt ChromeDriver bằng webdriver_manager.

2. **Danh sách URL và bảng**

- Một danh sách stat_urls chứa các liên kết đến các trang thống kê:
    + Standard Stats: https://fbref.com/en/comps/9/2024-2025/stats/...
    + Goalkeeping: https://fbref.com/en/comps/9/2024-2025/keepers/...
    + Shooting, Passing, Goal Creation, Defense, Possession, Miscellaneous.
- Mỗi URL tương ứng với một ID bảng (table_identifiers): stats_standard, stats_keeper, v.v.

3. **Tải và phân tích trang**

- Selenium tải từng URL, chờ 3 giây để nội dung JavaScript được tải.
- BeautifulSoup phân tích HTML, tìm các bảng ẩn trong HTML comments (FBref lưu bảng trong bình luận để tối ưu hóa).
- Trích xuất bảng bằng cách tìm thẻ <table> với ID phù hợp.

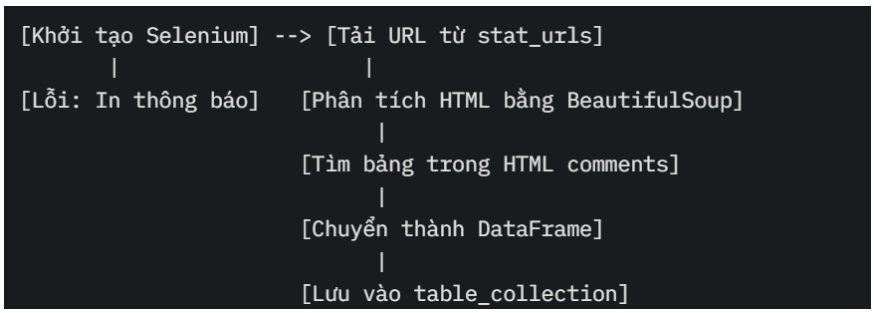
4. **Chuyển thành DataFrame**

- Sử dụng pd.read_html để chuyển bảng HTML thành DataFrame.
- Lưu vào từ điển table_collection với khóa là ID bảng.

5. **Xử lý lỗi**

- Try-except để xử lý lỗi khi tải trang, tìm bảng, hoặc đọc HTML.
- In thông báo lỗi (ví dụ: Table {table_id} not found!).

6. Mô tả sơ đồ luồng thu nhập dữ liệu



Giai đoạn 2: Xử lý và làm sạch dữ liệu

Mục đích: Chuẩn hóa dữ liệu (tên cột, tên cầu thủ, quốc gia, tuổi) và chuyển đổi kiểu dữ liệu.

Cách thực hiện

1. **Đổi tên cột**

- Sử dụng rename_columns_map để đổi tên cột không rõ ràng (ví dụ: Unnamed: 1 → Player, Performance → Gls).
- Loại bỏ cột trùng lặp bằng table_data.loc[:, ~table_data.columns.duplicated()].

2. **Chuẩn hóa văn bản**

- Tên cầu thủ (format_player_name):
  + Chuyển "Salah, Mohamed" thành "Mohamed Salah".
  + Loại bỏ khoảng trắng thừa, xử lý NaN thành "N/A".
- Quốc gia (get_country_code):
  + Trích xuất mã quốc gia (ví dụ: "eng ENG" → "ENG").
  + Điền "N/A" cho giá trị thiếu.
- Tuổi (parse_age_to_decimal):
  + Chuyển "25-100" thành 25.27 (năm + ngày/365), "25.5" thành 25.50.
  + Điền "N/A" cho giá trị không hợp lệ.

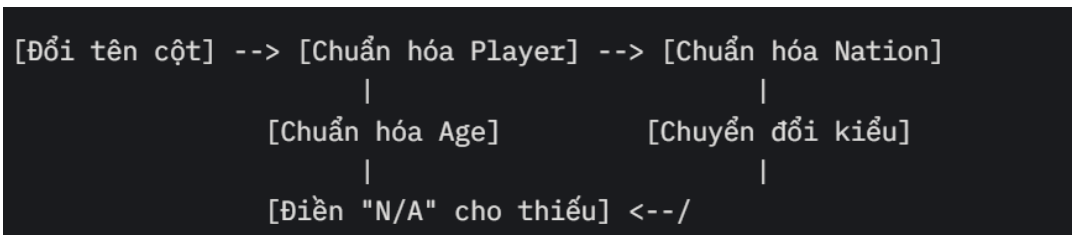
3. **Chuyển đổi kiểu dữ liệu**

- Cột số nguyên (Minutes, Gls, Ast, v.v.) → Int64 (hỗ trợ NaN).
- Cột số thực (xG, Gls per 90, Save%, v.v.) → float, làm tròn 2 chữ số.
- Cột văn bản (Player, Nation, Team, Position) → điền "N/A" cho giá trị thiếu.

4. **In kiểm tra**

- In danh sách cột, mẫu dữ liệu (Player, Age) để debug.

5. Sơ đồ xử lý dữ liệu



Giai đoạn 3: Gộp dữ liệu

Mục đích: Kết hợp các bảng thành một DataFrame tổng hợp.

Cách thực hiện

1. **Chọn cột mục tiêu**

- Chỉ giữ cột trong target_columns (bao gồm tất cả chỉ số yêu cầu).

2. **Loại bỏ trùng lặp**

- Loại bản ghi trùng lặp dựa trên Player bằng drop_duplicates.

3. **Gộp bảng**

- Sử dụng pd.merge với phương thức outer theo cột Player.
- Thêm validate="1:1" để đảm bảo mỗi cầu thủ chỉ xuất hiện một lần.
- Xử lý lỗi gộp bằng try-except.

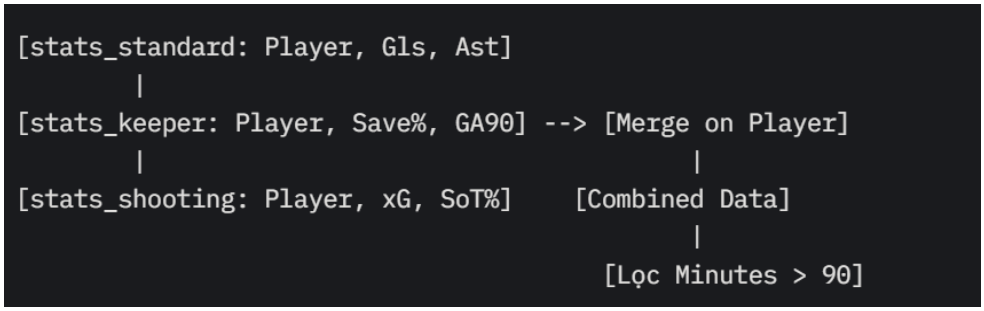
4. **Sắp xếp cột**

- Giữ các cột trong target_columns theo thứ tự xác định.

5. **Lọc cầu thủ**

- Sau gộp, lọc cầu thủ có Minutes không phải NaN và Minutes > 90.

6. Sơ đồ gộp dữ liệu



Giai đoạn 4: Lưu kết quả

Mục đích: Lưu DataFrame tổng hợp vào results.csv với cấu trúc yêu cầu.

Cách thực hiện

1. **Lưu DataFrame**

- Lưu combined_data vào results.csv bằng to_csv với:
  + Encoding: utf-8-sig (hỗ trợ ký tự đặc biệt).
  + na_rep="N/A" để chuyển NaN thành "N/A".
  + Không lưu chỉ số (index=False).

2. **Các bước chuẩn bị trước đó**

- Đã lọc Minutes > 90.
- Đã điền "N/A" cho cột văn bản.
- Cột số giữ NaN, được chuyển thành "N/A" khi lưu.

3. Sơ đồ lưu kết quả

1.3.2 Giải thích cách thuật toán hoạt động giải quyết vấn đề 2

1. Xác định top 3 cao nhất và thấp nhất

1.1 Chuẩn bị dữ liệu

- Sử dụng DataFrame combined_data từ giai đoạn trước (đã lọc cầu thủ chơi trên 90 phút).
- Sao chép thành calc_data để xử lý, đảm bảo không thay đổi dữ liệu gốc.
- Chuyển đổi tất cả cột số thành kiểu float bằng pd.to_numeric(errors="coerce"), điền 0 cho giá trị thiếu (NaN).

1.2 Xác định cột số

- Tạo danh sách numeric_columns bằng cách loại bỏ các cột văn bản (Player, Nation, Team, Position).

1.3 Tính toán top 3

- Bước thực hiện
  + Khởi tạo từ điển stat_rankings để lưu kết quả.
  + Lặp qua từng cột trong numeric_columns.
  + Top 3 cao nhất:
    - Sắp xếp calc_data[["Player", "Team", col]] theo cột giảm dần (sort_values(by=col, ascending=False)).
    - Lấy 3 hàng đầu tiên bằng head(3).
    - Đổi tên cột thành "Value" và thêm cột "Rank" với giá trị ["1st", "2nd", "3rd"].
    - Lưu vào stat_rankings[col]["Highest"].
  + Top 3 thấp nhất:
    - Kiểm tra nếu tất cả giá trị trong cột là 0 (calc_data[col].eq(0).all()):
    - Nếu đúng, sắp xếp tăng dần và lấy 3 hàng đầu tiên (để xử lý các chỉ số như Save% cho thủ môn không ra sân).
    - Nếu không, lọc các giá trị lớn hơn 0 (non_zero_data = calc_data[calc_data[col] > 0]) để tránh xếp hạng các cầu thủ không có đóng góp (ví dụ: không ghi bàn cho Gls).
    - Sắp xếp tăng dần, lấy 3 hàng đầu tiên, đổi tên cột và thêm "Rank".
    - Lưu vào stat_rankings[col]["Lowest"].

- Lưu kết quả vào từ điển stat_rankings với cấu trúc:

```
stat_rankings = {
    "Gls": {
        "Highest": DataFrame[Rank, Player, Team, Value],
        "Lowest": DataFrame[Rank, Player, Team, Value]
    },
    ...
}
```

1.4. Lưu kết quả vào top_3.txt

- Bước thực hiện
  + Mở file top_3.txt ở chế độ ghi (w, encoding utf-8).
  + Lặp qua stat_rankings, ghi từng chỉ số theo định dạng:

```
Statistic: Gls
Top 3 Highest:
Rank Player Team Value
1st  Mohamed Salah  Liverpool  10
...
Top 3 Lowest:
Rank Player Team Value
1st  John Smith     Everton    1
...
------------------------------
```

  + Sử dụng to_string(index=False) để định dạng DataFrame thành văn bản.

1.5 Sơ đồ xếp hạng

```
[Calc Data] --> [Lặp qua numeric_columns]
                       |
               [Top 3 cao: sort descending]
                       |
               [Top 3 thấp: filter > 0, sort ascending]
                       |
               [Lưu vào stat_rankings]
                       |
               [Ghi vào top_3.txt]
```

2. Tính toán thống kê (Median, Mean, Std)

2.1 Chuẩn bị dữ liệu

- Bước thực hiện
  + Sử dụng calc_data (đã chuyển cột số thành float, điền 0 cho NaN).
  + Tạo danh sách teams bằng cách lấy giá trị duy nhất từ cột Team (sorted(calc_data["Team"].unique())).

2.2 Tính toán thống kê

- Bước thực hiện
  + Khởi tạo danh sách stat_rows để lưu kết quả.
  + Toàn giải đấu:
  + Tạo từ điển league_stats với khóa "" là "all".
  + Lặp qua numeric_columns, tính:

```
league_stats[f"Median of {col}"] = calc_data[col].median()
league_stats[f"Mean of {col}"] = calc_data[col].mean()
league_stats[f"Std of {col}"] = calc_data[col].std()
```

  + Thêm league_stats vào stat_rows.
- Lặp qua từng đội bóng
  + Lặp qua teams
    - Lọc dữ liệu cho đội (team_subset = calc_data[calc_data["Team"] == team]).
    - Tạo từ điển team_metrics với khóa "" là tên đội.
    - Tính median, mean, std cho mỗi cột số, tương tự toàn giải.
    - Thêm team_metrics vào stat_rows.

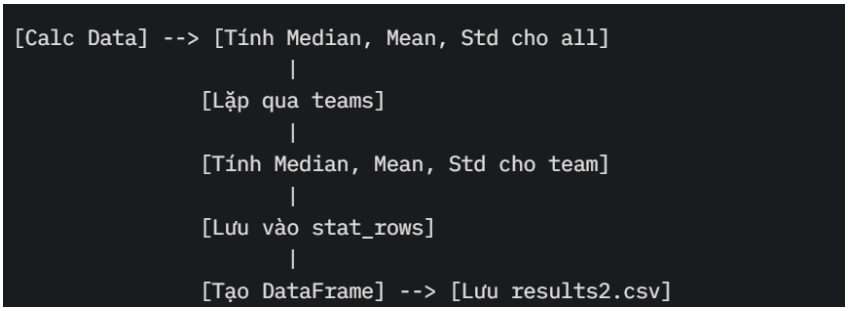
2.3 Tạo và lưu DataFrame

- Bước thực hiện
  + Chuyển stat_rows thành DataFrame stats_summary bằng pd.DataFrame(stat_rows).
  + Đổi tên cột "" thành chuỗi rỗng ("") để khớp định dạng yêu cầu.
  + Làm tròn tất cả cột số đến 2 chữ số thập phân (stats_summary[col].round(2)).
  + Lưu vào results2.csv bằng to_csv với:
    - Encoding: utf-8-sig (hỗ trợ ký tự đặc biệt).
    - Không lưu chỉ số (index=False).

2.4 Lưu vào results2.csv

- Lưu stats_summary bằng to_csv với:
  + Encoding: utf-8-sig.
  + Không lưu chỉ số (index=False).

2.5 Sơ đồ tính toán thống kê



3. Vẽ histogram phân bố chỉ số

3.1 Chuẩn bị dữ liệu

- Bước thực hiện
  + Định nghĩa danh sách plot_stats với 6 chỉ số: Gls per 90, xG per 90, SCA90, GA90, TklW, Blocks.
  + Tạo thư mục lưu trữ bằng os.makedirs:
    - histograms/league cho toàn giải.
    - histograms/teams cho từng đội.
  + Lấy danh sách teams từ calc_data["Team"].unique().

3.2 Vẽ histogram cho toàn giải:

- Bước thực hiện
  + Lặp qua plot_stats
    - Kiểm tra nếu chỉ số tồn tại trong calc_data, bỏ qua nếu không (if stat not in calc_data.columns).
    - Tạo figure mới (plt.figure(figsize=(10, 6))).
    - Vẽ histogram bằng plt.hist:
      - Dữ liệu: calc_data[stat].
      - Số bin: 20 (cho phân bố chi tiết).
      - Màu: skyblue, viền đen (edgecolor="black").
    - Tùy chỉnh:
      - Tiêu đề: League-Wide Distribution of {stat}.
      - Nhãn trục X: Tên chỉ số (stat).
      - Nhãn trục Y: Number of Players.
      - Lưới: grid(True, alpha=0.3) cho độ trong suốt.
    - Lưu vào histograms/league/{stat}_league.png với bbox_inches="tight" (tránh cắt nội dung).
    - Đóng figure (plt.close()) để giải phóng bộ nhớ.

3.3 Vẽ histogram cho từng đội:

- Bước thực hiện:

+ Lặp qua teams và plot_stats:
    - Lọc dữ liệu cho đội (team_subset = calc_data[calc_data["Team"] == team]).
    - Tạo figure mới (plt.figure(figsize=(8, 6))).
    - Vẽ histogram:
        - Dữ liệu: team_subset[stat].
        - Số bin: 10 (ít hơn vì số cầu thủ mỗi đội nhỏ).
        - Màu:
            - lightgreen cho chỉ số phòng ngự (GA90, TklW, Blocks) để phân biệt.
            - skyblue cho các chỉ số khác.
            - Độ trong suốt: alpha=0.7.
    - Tùy chỉnh tương tự, với tiêu đề: {team} - Distribution of {stat}.
    - Lưu vào histograms/teams/{team}_{stat_file_name}.png (thay khoảng trắng trong stat bằng _).
    - Đóng figure.

- Hình ảnh minh họa:
    + Histogram mẫu:
        - Mô tả: Histogram cho Gls per 90:
            - Trục X: Giá trị (0 đến 1.5).
            - Trục Y: Số cầu thủ.
            - Tiêu đề: League-Wide Distribution of Gls per 90.
            - Màu: skyblue, viền đen.
        - Cách tạo: Chạy code hoặc giả lập:

```
import matplotlib.pyplot as plt
data = [0.2, 0.5, 0.3, 1.0, 0.0, 0.8, 0.4]
plt.hist(data, bins=20, color="skyblue", edgecolor="black")
plt.title("League-Wide Distribution of Gls per 90")
plt.xlabel("Gls per 90")
plt.ylabel("Number of Players")
plt.grid(True, alpha=0.3)
plt.savefig("histogram_example.png")
plt.close()
```

3.4 Cây thư mục histograms

```
histograms/
├── league/
│   ├── Gls_per_90_league.png
│   ├── xG_per_90_league.png
│   ...
├── teams/
│   ├── Arsenal_Gls_per_90.png
│   ├── Liverpool_Gls_per_90.png
│   ...
```

4. Xác định đội bóng xuất sắc

4.1 Tính trung bình theo đội

- Bước thực hiện
  + Sử dụng calc_data.groupby("Team")[numeric_columns].mean() để tính trung bình mỗi chỉ số cho từng đội.
  + Kết quả là DataFrame team_averages
    - Cột: Các chỉ số số (Gls, xG, Save%, v.v.).
    - Hàng: Các đội (Arsenal, Liverpool, v.v.).

4.2 Tìm đội dẫn đầu mỗi chỉ số

- Bước thực hiện
  + Khởi tạo danh sách top_team_stats.
  + Lặp qua numeric_columns:
    - Tìm chỉ số lớn nhất (idxmax()) trong cột của team_averages.
    - Lấy tên đội (top_row["Team"]) và giá trị trung bình (top_row[stat]).
    - Làm tròn giá trị đến 2 chữ số thập phân.
    - Thêm vào top_team_stats với cấu trúc:

```
{"Statistic": stat, "Team": team, "Mean Value": value}
```

    - Chuyển top_team_stats thành DataFrame top_team_data.

4.3 Lưu vào highest_team_stats.csv

- Bước thực hiện:
  + Lưu top_team_data bằng to_csv với:
    - Cột: Statistic, Team, Mean Value.
    - Encoding: utf-8-sig.
    - Không lưu chỉ số.

4.4 Xác định đội tốt nhất

- Bước thực hiện:
  + Đọc highest_team_stats.csv vào DataFrame.
  + Định nghĩa danh sách negative_metrics chứa các chỉ số tiêu cực:

```
negative_metrics = ["GA90", "CrdY", "CrdR", "Lost", "Mis", "Dis", "Fls",
"Off", "Aerl Lost"]
```

  + Lọc các chỉ số tích cực (~top_team_data["Statistic"].isin(negative_metrics)).
  + Đếm số lần mỗi đội xuất hiện trong các chỉ số tích cực (team_rank_counts = positive_stats_data["Team"].value_counts()).
  + Chọn đội có số lần xuất hiện nhiều nhất (top_team = team_rank_counts.idxmax()).
  + Lấy số lượng chỉ số dẫn đầu (lead_count = team_rank_counts.max()).
  + In kết quả:

```
print(f"The best-performing team ... is: {top_team}")
print(f"They lead in {lead_count} out of {len(positive_stats_data)} positive
statistics.")
```

4.5 Sơ đồ xác định đội tốt nhất

- Mô tả bằng Flowchart:

```
[Calc Data] --> [Groupby Team, tính mean]
     |
[Tìm đội có max mean mỗi chỉ số]
     |
[Lưu vào highest_team_stats.csv]
     |
[Lọc chỉ số tích cực]
     |
[Đếm số lần dẫn đầu mỗi đội]
     |
[Chọn đội có số lần cao nhất]
     |
[In kết quả]
```

2. Vấn đề 3

2.1 Thuật toán K-means và nhận xét về cách chia nhóm

2.1.1 Đọc và chuẩn bị dữ liệu

1. **Đọc file CSV**

- **Code:** data = pd.read_csv('result.csv')
- **Giải thích:**
    + Tải file result.csv (được tạo từ yêu cầu trước, chứa dữ liệu thống kê cầu thủ mùa giải Premier League 2024-2025).
    + File chứa các cột như Player, Nation, Team, Position, và các chỉ số thống kê (Gls per 90, xG per 90, Tkl, v.v.).
    + Dữ liệu được lưu vào DataFrame data để xử lý tiếp theo.

2. **Chọn đặc trưng:**

- **Code:**

```
features = [
    'Gls per 90', 'Ast per 90', 'xG per 90', 'xAG per 90', 'SCA90', 'GCA90',
    'Cmp%', 'TotDist', 'KP', 'PPA', 'Tkl', 'Blocks', 'Int',
    'Touches', 'PrgC', 'PrgP', 'PrgR', 'Carries'
]
```

- **Giải thích:**
    + Lựa chọn 18 đặc trưng (features) đại diện cho hiệu suất cầu thủ, bao gồm:
        - Tấn công: Gls per 90, Ast per 90, xG per 90, xAG per 90, SCA90, GCA90 (bàn thắng, kiến tạo, và tạo cơ hội mỗi 90 phút).
        - Chuyền bóng: Cmp% (tỷ lệ hoàn thành đường chuyền), TotDist (tổng khoảng cách chuyền bóng tiến bộ), KP (đường chuyền then chốt), PPA (đường chuyền vào khu vực penalty).
        - Phòng ngự: Tkl (tắc bóng), Blocks (chặn bóng), Int (đánh chặn).
        - Tham gia: Touches (lần chạm bóng), PrgC (dẫn bóng tiến bộ), PrgP (chuyền bóng tiến bộ), PrgR (nhận bóng tiến bộ), Carries (lần mang bóng).
    + Các đặc trưng này được chọn để phản ánh các khía cạnh khác nhau của hiệu suất cầu thủ, từ tấn công, phòng ngự đến tham gia vào lối chơi.

3. **Trích xuất dữ liệu đặc trưng**

- **Code:** X = data[features].copy()

- **Giải thích:**
    + Tạo DataFrame X chỉ chứa các cột trong features.
    + Sử dụng .copy() để tạo bản sao, tránh thay đổi dữ liệu gốc trong data.

2.1.2 Xử lý dữ liệu thiếu

1. **Thay thế 'N/A'**

- **Code:** X = X.replace('N/A', 0)
- **Giải thích:**
    + Thay tất cả giá trị 'N/A' (chuỗi văn bản) trong X bằng 0.
    + Điều này cần thiết vì result.csv có thể chứa 'N/A' cho các chỉ số không áp dụng (ví dụ: Save% cho cầu thủ không phải thủ môn).

2. **Điền giá trị NaN**

- **Code:** X = X.fillna(0)
- **Giải thích:**
    + Điền tất cả giá trị NaN (giá trị thiếu) bằng 0.
    + NaN có thể xuất hiện do dữ liệu không được ghi nhận hoặc lỗi xử lý trước đó.

2.1.3 Chuẩn hóa dữ liệu

1. **Khởi tạo StandardScaler**

- **Code:** scaler = StandardScaler()
- **Giải thích:**
    + Tạo đối tượng StandardScaler từ sklearn.preprocessing để chuẩn hóa dữ liệu.
    + StandardScaler chuyển đổi dữ liệu sao cho mỗi đặc trưng có trung bình = 0 và độ lệch chuẩn = 1, giúp các đặc trưng có cùng tầm quan trọng trong K-means.

2. **Chuẩn hóa đặc trưng:**

- **Code:** X_scaled = scaler.fit_transform(X)
- **Giải thích:**
    + fit_transform tính toán trung bình và độ lệch chuẩn của X, sau đó chuẩn hóa:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

    + với μ là trung bình và σ là độ lệch chuẩn của mỗi cột.
    + Kết quả là X_scaled, một mảng NumPy chứa dữ liệu chuẩn hóa.

2.1.4 Tính toán inertia và chọn số cụm

1. **Tính inertia cho các giá trị k:**

- **Code:**

```
inertia = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
```

- **Giải thích:**
    + Khởi tạo danh sách inertia để lưu giá trị WCSS (Within-Cluster Sum of Squares).
    + Thử các giá trị k từ 1 đến 10 (k_range).
    + Với mỗi k:
        - Tạo mô hình K-means với:
            - n_clusters=k: Số cụm.
            - random_state=42: Đảm bảo kết quả lặp lại được.
            - n_init=10: Chạy K-means 10 lần với các điểm khởi tạo ngẫu nhiên khác nhau, chọn kết quả tốt nhất.
        - Huấn luyện mô hình trên X_scaled (kmeans.fit(X_scaled)).
        - Lấy giá trị inertia (kmeans.inertia_), tức là tổng bình phương khoảng cách từ mỗi điểm đến tâm cụm gần nhất.
    + Lưu inertia cho từng k vào danh sách inertia.

2. **Chọn giá trị k:**

- **Code:** Trong đoạn code đã chọn k=4 (tên biến: chosen_k)
- **Giải thích:**
    + Chọn k=4 làm số cụm tối ưu (dựa trên phân tích elbow plot, như giải thích trong nhận xét).
    + Giá trị này được cố định để vẽ điểm nổi bật trên biểu đồ.

2.1.5 Vẽ elbow plot

1. **Tạo biểu đồ:**

- **Code:**

```
plt.figure(figsize=(8, 6))
plt.plot(k_range, inertia, marker='o', linestyle='-', color='b',
label='Inertia')
plt.scatter([chosen_k], [inertia[chosen_k-1]], color='red', s=100, label=f'k=
{chosen_k}', zorder=5)
```

- **Giải thích:**

+ Tạo figure kích thước 8x6 inch.
+ Vẽ đường nối các điểm (k, inertia) với:
    - marker='o': Đánh dấu điểm bằng hình tròn.
    - linestyle='-': Đường liền nét.
    - color='b': Màu xanh dương.
    - label='Inertia': Nhãn cho đường.

- Đánh dấu điểm k=4 bằng chấm đỏ (color='red', kích thước s=100), đặt zorder=5 để đảm bảo điểm nằm trên cùng.

2. **Tùy chỉnh biểu đồ:**

- **Code:**

```
plt.title(f'Elbow Plot (k={chosen_k})')
plt.xlabel(f'Number of Clusters (k = {chosen_k})')
plt.ylabel('Inertia')
plt.grid(True)
plt.xticks(k_range)
plt.legend()
```

- **Giải thích:**

+ Tiêu đề: Elbow Plot (k=4), hiển thị giá trị k được chọn.
+ Nhãn trục X: Number of Clusters (k = 4), nhấn mạnh k=4.
+ Nhãn trục Y: Inertia.
+ Thêm lưới (grid(True)).
+ Đặt các điểm trên trục X là k_range (1, 2, ..., 10).
+ Thêm chú thích (legend) để hiển thị nhãn Inertia và k=4.

3. **Lưu và đóng biểu đồ:**

- **Code:**

```
plt.savefig('elbow_plot.png', format='png', dpi=300, bbox_inches='tight')
plt.close()
print(f"Elbow plot with k={chosen_k} has been saved as 'elbow_plot.png'.")
```

- **Giải thích:**

+ Lưu biểu đồ thành file elbow plot.png với độ phân giải cao (dpi=300) và bbox_inches="tight" để tránh cắt nội dung.
+ Đóng figure (plt.close()) để giải phóng bộ nhớ.
+ In thông báo xác nhận lưu file.

2.1.6 Sơ đồ tổng quát

```
[Start] (Oval)
   |
   v
[Đọc result.csv, tạo DataFrame data] (Rectangle)
   |
   v
[Chọn 18 features, tạo X = data[features].copy()] (Rectangle)
   |
   v
[Xử lý dữ liệu thiếu: X.replace('N/A', 0), X.fillna(0)] (Rectangle)
   |
   v
[Chuẩn hóa dữ liệu: scaler = StandardScaler(), X_scaled = scaler.fit_transform(X)]
(Rectangle)
   |
   v
[Khởi tạo inertia = [], k_range = range(1, 11)] (Rectangle)
   |
   v
[For k in k_range] (Diamond - Loop Start)
   |
   v
   [Tạo KMeans(n_clusters=k, random_state=42, n_init=10)] (Rectangle)
   |
   v
   [Fit KMeans trên X_scaled, tính inertia, thêm vào inertia list] (Rectangle)
   |
   v
[End Loop] (Diamond - Loop End)
   |
   v
[Chọn chosen_k = 4] (Rectangle)
   |
   v
[Vẽ elbow plot: plt.plot(k_range, inertia), đánh dấu k=4] (Rectangle)
   |
   v
[Tùy chỉnh plot: title, xlabel, ylabel, grid, legend] (Rectangle)
   |
   v
[Lưu plot thành elbow_plot.png, đóng plt] (Rectangle)
   |
   v
[In thông báo: "Elbow plot with k=4 has been saved..."] (Rectangle)
   |
   v
[End] (Oval)
```

2.1.7 Nhận xét về số lượng nhóm và kết quả

- **Cách code hỗ trợ:**
  - + Code không thực hiện phân cụm cuối cùng (chỉ dừng ở vẽ elbow plot), nhưng cung cấp dữ liệu để chọn k=4 thông qua elbow plot.
  - + Nhận xét trong code (bằng tiếng Việt) giải thích lý do chọn k=4 và đánh giá kết quả.
- **Nhận xét trong code:**
  - + Số lượng nhóm: Chọn k=4 vì:
    - Elbow point: Inertia giảm mạnh từ k=1 đến k=4, sau đó giảm chậm, cho thấy k=4 là điểm cân bằng giữa độ chặt chẽ cụm và tính đơn giản.

- Ý nghĩa bóng đá: 4 nhóm có thể đại diện cho các vai trò:
    - Nhóm 1: Tiền đạo (cao Gls per 90, xG per 90).
    - Nhóm 2: Tiền vệ sáng tạo (cao Ast per 90, KP).
    - Nhóm 3: Hậu vệ phòng ngự (cao Tkl, Blocks).
    - Nhóm 4: Cầu thủ đa năng hoặc thủ môn (nếu không lọc).
- Tính thực tiễn: 4 nhóm dễ diễn giải và áp dụng.

- **Nhận xét kết quả:**
    + Hiệu quả: 18 đặc trưng phản ánh tốt các vai trò cầu thủ.
    + Thủ môn: Có thể gây nhiễu nếu không lọc (vì thiếu chỉ số như Save%).
    + Hạn chế: Cầu thủ ít phút thi đấu có thể ảnh hưởng đến phân cụm.
    + Ứng dụng: Phân cụm giúp phân tích đội hình, so sánh cầu thủ, hoặc chuyển nhượng.

2.2 Thuật toán sử dụng PCA

Chú ý: Thuật toán sử dụng PCA được phát triển dựa trên thuật toán K-means vậy nên ta sẽ phân tích điểm khác biệt (phần sử dụng PCA) (từ dòng 29 trong file code).

2.2.1 Chuẩn hóa dữ liệu

1. **Khởi tạo StandardScaler:**

- **Code**: scaler = StandardScaler()
- **Giải thích**:
    + Tạo đối tượng StandardScaler từ sklearn.preprocessing.
    + StandardScaler chuẩn hóa dữ liệu sao cho mỗi đặc trưng có trung bình = 0 và độ lệch chuẩn = 1, đảm bảo các đặc trưng có cùng tầm quan trọng.

2. **Chuẩn hóa đặc trưng:**

- **Code**: X_scaled = scaler.fit_transform(X)
- **Giải thích**:
    + fit_transform tính toán trung bình (μ\muμ) và độ lệch chuẩn (σ\sigmaσ) của mỗi cột trong X, sau đó chuẩn hóa:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

+ Kết quả là X_scaled, một mảng NumPy chứa dữ liệu chuẩn hóa (18 đặc trưng, mỗi đặc trưng có trung bình = 0, độ lệch chuẩn = 1).

2.2.2 Giảm chiều dữ liệu bằng PCA

1. **Khởi tạo PCA:**

- **Code**: pca = PCA(n_components=2)
- **Giải thích**:
  + Tạo đối tượng PCA từ sklearn.decomposition với n_components=2, yêu cầu giảm dữ liệu xuống 2 chiều.
  + PCA tìm các **thành phần chính** (principal components) là các hướng có phương sai lớn nhất trong dữ liệu, sau đó chiếu dữ liệu lên 2 thành phần đầu tiên.

2. **Áp dụng PCA:**

- **Code**: X_pca = pca.fit_transform(X_scaled)
- **Giải thích**:
  + fit_transform thực hiện các bước:
    - Tính ma trận hiệp phương sai của X_scaled.
    - Tìm các vector riêng (eigenvectors) và giá trị riêng (eigenvalues) để xác định 2 thành phần chính có phương sai lớn nhất.
  + Chiếu dữ liệu X_scaled (18 chiều) lên không gian 2 chiều của 2 thành phần chính.
  + Kết quả là X_pca, một mảng NumPy với shape (n_samples, 2), mỗi hàng là tọa độ của một cầu thủ trên 2 thành phần chính.

2.2.3 Phân cụm bằng K-means

1. **Khởi tạo K-means:**

- **Code**: kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
- **Giải thích**:
  + Tạo mô hình K-means với:
    - n_clusters=4: Số cụm là 4 (dựa trên elbow plot từ yêu cầu trước).
    - random_state=42: Đảm bảo kết quả lặp lại được.
    - n_init=10: Chạy K-means 10 lần với các điểm khởi tạo ngẫu nhiên khác nhau, chọn kết quả có inertia thấp nhất.

2. **Phân cụm:**

- **Code**: cluster_labels = kmeans.fit_predict(X_pca)

- **Giải thích**:
  + fit_predict thực hiện:
    - Huấn luyện K-means trên X_pca (dữ liệu 2D).
    - Gán mỗi cầu thủ vào một cụm (0, 1, 2, hoặc 3).
  + Kết quả là cluster_labels, một mảng NumPy chứa nhãn cụm (0 đến 3) cho mỗi cầu thủ.

2.2.4 Vẽ biểu đồ phân cụm 2D

1. **Tạo biểu đồ phân tán**

- **Code**:

```
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1],
c=cluster_labels, cmap='viridis', s=50)
```

- **Giải thích**:
  + Tạo figure kích thước 8x6 inch.
  + Vẽ scatter plot với:
    - X_pca[:, 0]: Tọa độ PC1 (trục X).
    - X_pca[:, 1]: Tọa độ PC2 (trục Y).
    - c=cluster_labels: Tô màu theo nhãn cụm (0, 1, 2, 3).
    - cmap='viridis': Sử dụng bảng màu viridis (từ xanh đến vàng).
    - s=50: Kích thước điểm là 50.
    - Lưu đối tượng scatter để thêm colorbar.

2. **Tùy chỉnh biểu đồ**

- **Code**:

```
plt.title('2D Cluster Plot of Players (PCA, k=4)')
plt.xlabel('Forward and Attacking Midfielder Ability')
plt.ylabel('Defender and Defensive Midfielder Ability')
plt.colorbar(scatter, label='Cluster')
plt.grid(True)
```

- **Giải thích**:
  + Tiêu đề: 2D Cluster Plot of Players (PCA, k=4), chỉ rõ phương pháp (PCA) và số cụm (k=4).
  + Nhãn trục X: Forward and Attacking Midfielder Ability, giả định PC1 liên quan đến khả năng tấn công (dựa trên các đặc trưng như Gls per 90, Ast per 90).

+ Nhãn trục Y: Defender and Defensive Midfielder Ability, giả định PC2 liên quan đến khả năng phòng ngự (dựa trên Tkl, Blocks).
+ Thêm colorbar để hiển thị nhãn cụm (0 đến 3).
+ Thêm lưới (grid(True)) để dễ đọc.

3. **Lưu và đóng biểu đồ:**

- **Code**:

```
plt.savefig('cluster_plot.png', format='png', dpi=300,
bbox_inches='tight')
plt.close()
print("2D cluster plot has been saved as
'cluster_plot.png'.")
```

- **Giải thích**:
  + Lưu biểu đồ thành file cluster_plot.png với độ phân giải cao (dpi=300) và bbox_inches="tight" để tránh cắt nội dung.
  + Đóng figure (plt.close()) để giải phóng bộ nhớ.
  + In thông báo xác nhận lưu file.

2.2.5 Sơ đồ tổng thể thuật toán

```
[Đọc result.csv] --> [Chọn features, tạo X]
        |
[Xử lý 'N/A', NaN -> 0]
        |
[Chuẩn hóa X -> X_scaled]
        |
[PCA: Giảm chiều -> X_pca (2D)]
        |
[K-means: Phân cụm k=4 -> cluster_labels]
        |
[Vẽ scatter plot với X_pca, màu theo cluster_labels]
        |
[Lưu cluster_plot.png]
```

3. Vấn đề 4

3.1 Thu thập giá trị chuyển nhượng của cầu thủ chơi trên 900 phút mùa giải 2024-2025

3.1.1 Thiết lập môi trường và kiểm tra file

- **Code:**

```
root_dir = r"C:\Users\DD\OneDrive\Documents\newfolder(2)\btlpython"
csv_folder = os.path.join(root_dir, "csv")
os.makedirs(csv_folder, exist_ok=True)
result_csv_path = os.path.join(csv_folder, "result.csv")

if not os.path.exists(result_csv_path):
    print(f"Error: File {result_csv_path} does not exist.")
    exit()

try:
    data_frame = pd.read_csv(result_csv_path, na_values=["N/A"])
except Exception as e:
    print(f"Error reading {result_csv_path}: {str(e)}")
    exit()
```

- **Cách hoạt động:**
  + **Xác định thư mục:**
    - root_dir: Thư mục gốc chứa dự án (C:\Users\DD\...).
    - csv_folder: Thư mục con csv để lưu các file CSV, được tạo bằng os.makedirs(csv_folder, exist_ok=True) nếu chưa tồn tại.
    - result_csv_path: Đường dẫn đến file result.csv, chứa dữ liệu thống kê cầu thủ (như tên, số phút thi đấu, bàn thắng, v.v.).
  + **Kiểm tra file:**
    - Kiểm tra xem result.csv có tồn tại không bằng os.path.exists. Nếu không, in lỗi và thoát chương trình.
  + **Đọc file CSV:**
    - pd.read_csv(result_csv_path, na_values=["N/A"]) đọc result.csv vào DataFrame data_frame, chuyển các giá trị "N/A" thành NaN.
    - Được bao bọc trong khối try-except để xử lý lỗi (như file bị hỏng hoặc định dạng sai).

3.1.2 Lọc cầu thủ trên 900 phút

- **Code:**

```
calc_data_frame = data_frame.copy()
filtered_data_frame = calc_data_frame[calc_data_frame['Minutes'] > 900].copy()
print(f"Number of players with more than 900 minutes: {len(filtered_data_frame)}")

filtered_csv_path = os.path.join(root_dir, "csv", "players_over_900_minutes.csv")
filtered_data_frame.to_csv(filtered_csv_path, index=False, encoding='utf-8-sig')
print(f"Saved filtered players to {filtered_csv_path} with {filtered_data_frame.shape[0]
```

- **Cách hoạt động**
  + Sao chép dữ liệu:

- calc_data_frame = data_frame.copy() tạo bản sao của data_frame để tránh thay đổi dữ liệu gốc.

+ Lọc cầu thủ:
  - filtered_data_frame = calc_data_frame[calc_data_frame['Minutes'] > 900].copy() lọc các hàng có giá trị cột Minutes lớn hơn 900.
  - .copy() đảm bảo DataFrame mới độc lập.
+ In thông tin:
  - In số lượng cầu thủ được lọc (len(filtered_data_frame)).
+ Lưu file:
  - Lưu DataFrame đã lọc vào players_over_900_minutes.csv bằng to_csv.
  - index=False: Không lưu chỉ số hàng.
  - encoding='utf-8-sig': Hỗ trợ ký tự đặc biệt.
  - In đường dẫn file và kích thước DataFrame (shape[0] là số hàng, shape[1] là số cột).

3.1.3. Chuẩn bị tên cầu thủ và hàm rút ngắn tên

- **Code:**

```
def truncate_name(name):
    parts = name.strip().split()
    return " ".join(parts[:2]) if len(parts) >= 2 else name

filtered_csv_file = os.path.join(root_dir, "csv", "players_over_900_minutes.csv")
try:
    players_data_frame = pd.read_csv(filtered_csv_file)
except Exception as e:
    print(f"Error reading {filtered_csv_file}: {str(e)}")
    exit()

short_player_names = [truncate_name(name) for name in
players_data_frame['Player'].str.strip()]
minutes_by_player = dict(zip(players_data_frame['Player'].str.strip(),
players_data_frame['Minutes']))
```

- **Cách hoạt động:**

+ Hàm truncate_name:
  - Nhận vào tên cầu thủ (name), loại bỏ khoảng trắng thừa bằng strip().
  - Chia tên thành các phần (dựa trên khoảng trắng) bằng split().
  - Nếu tên có từ 2 phần trở lên, trả về 2 phần đầu tiên (ví dụ: "Mohamed Salah Eldin" → "Mohamed Salah").
  - Nếu tên chỉ có 1 phần, trả về nguyên tên.
+ Đọc file đã lọc:
  - Đọc players_over_900_minutes.csv vào players_data_frame.

- Xử lý lỗi đọc file bằng try-except.

+ Rút ngắn tên cầu thủ:
  - players_data_frame['Player'].str.strip() loại bỏ khoảng trắng thừa trong cột Player.
  - [truncate_name(name) for name in ...] áp dụng truncate_name cho từng tên, tạo danh sách short_player_names.
+ Tạo từ điển thời gian thi đấu:
  - dict(zip(players_data_frame['Player'].str.strip(), players_data_frame['Minutes'])) tạo từ điển ánh xạ tên cầu thủ sang số phút thi đấu (dùng để kiểm tra sau nếu cần).

3.1.4. Cấu hình trình duyệt và cào dữ liệu chuyển nhượng

- **Code:**

```
browser_options = Options()
browser_options.add_argument("--headless")
browser_options.add_argument("--no-sandbox")
browser_options.add_argument("--disable-dev-shm-usage")
browser_options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36")

browser_driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=browser_options)
transfer_base_url = "https://www.footballtransfers.com/us/transfers/confirmed/2024-
2025/uk-premier-league/"
transfer_urls = [f"{transfer_base_url}{i}" for i in range(1, 15)]
transfer_data = []

try:
    for url in transfer_urls:
        print(f"Scraping: {url}")
        browser_driver.get(url)
        try:
            transfer_table = WebDriverWait(browser_driver, 20).until(
                EC.presence_of_element_located((By.CLASS_NAME, "transfer-table"))
            )
            table_rows = transfer_table.find_elements(By.TAG_NAME, "tr")
            print(f"Found {len(table_rows)} rows in table at {url}")
            ...
        except Exception as e:
            print(f"Error processing {url}: {str(e)}")
finally:
    browser_driver.quit()
```

- **Cách hoạt động:**

+ Cấu hình trình duyệt:
  - browser_options = Options() tạo đối tượng tùy chọn cho trình duyệt Chrome.
  - --headless: Chạy trình duyệt ở chế độ không giao diện (ẩn), tiết kiệm tài nguyên.

- --no-sandbox, --disable-dev-shm-usage: Tăng tính ổn định trên các hệ thống.
- user-agent: Giả lập trình duyệt Chrome để tránh bị chặn.

+ Khởi tạo WebDriver:
  - webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=browser_options) sử dụng webdriver-manager để tự động tải và cài đặt ChromeDriver.
+ Tạo danh sách URL:
  - transfer_base_url: URL gốc của trang chuyển nhượng Ngoại hạng Anh 2024-2025.
  - transfer_urls: Tạo danh sách 14 URL (từ trang 1 đến trang 14) để cào dữ liệu từ nhiều trang.
+ Cào dữ liệu:
  - Lặp qua từng URL trong transfer_urls.
  - browser_driver.get(url) truy cập trang web.
  - WebDriverWait(browser_driver, 20).until(...) chờ tối đa 20 giây để bảng chuyển nhượng (class transfer-table) xuất hiện.
  - table_rows = transfer_table.find_elements(By.TAG_NAME, "tr") lấy tất cả các hàng (tr) trong bảng.
+ Xử lý lỗi và đóng trình duyệt:
  - try-except xử lý lỗi khi truy cập hoặc lấy bảng (như trang không tải được).
  - finally: browser_driver.quit() đảm bảo trình duyệt đóng sau khi hoàn tất, giải phóng tài nguyên.

3.1.5 Xử lý dữ liệu bảng và khớp tên cầu thủ

- **Code:**

```
for row in table_rows:
    table_columns = row.find_elements(By.TAG_NAME, "td")
    if table_columns and len(table_columns) >= 2:
        full_player_name = table_columns[0].text.strip().split("\n")[0].strip()
        short_player_name = truncate_name(full_player_name)
        transfer_fee = table_columns[-1].text.strip() if len(table_columns) >= 3 else
"N/A"
        print(f"Processing player: {full_player_name}, Short name:
{short_player_name}, Fee: {transfer_fee}")
        top_match = process.extractOne(short_player_name, short_player_names,
scorer=fuzz.token_sort_ratio)
        if top_match and top_match[1] >= 80:
            matched_player_name = top_match[0]
            print(f"Matched: {full_player_name} -> {matched_player_name} (Score:
{top_match[1]})")
            transfer_data.append([full_player_name, transfer_fee])
    else:
        print(f"Skipping row with insufficient columns: {len(table_columns)}")
```

- **Cách hoạt động:**
  + Lấy dữ liệu từ bảng:
    - table_columns = row.find_elements(By.TAG_NAME, "td") lấy tất cả cột (td) trong mỗi hàng.
    - Kiểm tra table_columns tồn tại và có ít nhất 2 cột để tránh lỗi.
  + Trích xuất thông tin:
    - full_player_name: Lấy tên đầy đủ từ cột đầu tiên (table_columns[0]), loại bỏ khoảng trắng và dòng thừa (split("\n")[0]).
    - short_player_name = truncate_name(full_player_name) rút ngắn tên.
    - transfer_fee: Lấy phí chuyển nhượng từ cột cuối (table_columns[-1]), hoặc "N/A" nếu không đủ cột.
  + Khớp tên cầu thủ:
    - process.extractOne(short_player_name,short_player_names, scorer=fuzz.token_sort_ratio) sử dụng fuzzywuzzy để tìm tên khớp nhất trong short_player_names.
    - fuzz.token_sort_ratio: So sánh tên dựa trên các từ, bỏ qua thứ tự, phù hợp với tên cầu thủ (ví dụ: "Mohamed Salah" khớp với "Salah Mohamed").
    - Nếu độ khớp (top_match[1]) ≥ 80, thêm [full_player_name, transfer_fee] vào transfer_data.
  + In thông tin:
    - In tên đầy đủ, tên ngắn, phí chuyển nhượng, và kết quả khớp để theo dõi.

3.1.6 Lưu kết quả

- **Code:**

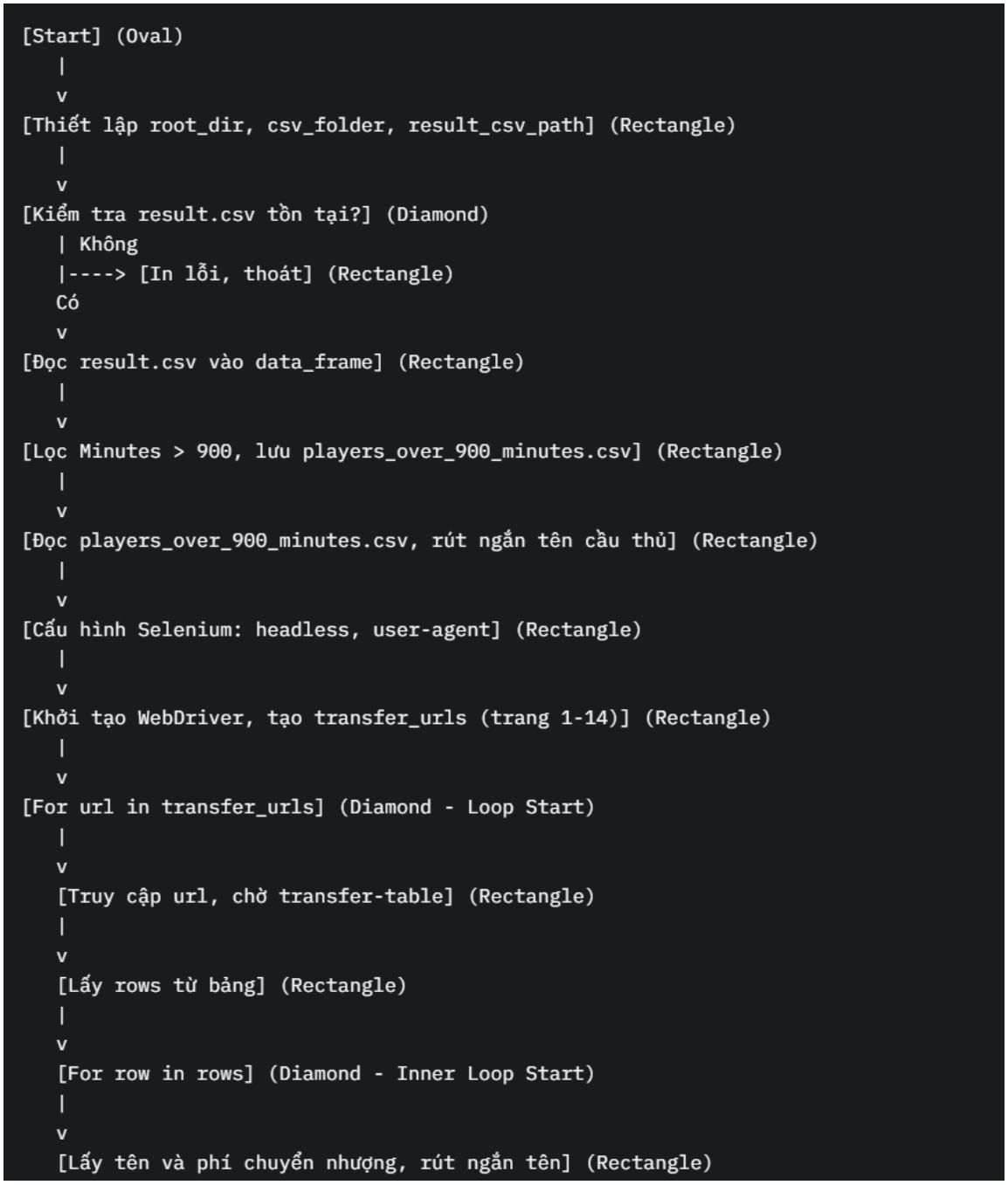
```
if transfer_data:
    transfer_data_frame = pd.DataFrame(transfer_data, columns=['Player', 'Price'])
    transfer_data_frame.to_csv(os.path.join(root_dir, "csv",
"player_transfer_fee.csv"), index=False)
    print(f"Results saved to '{os.path.join(root_dir, 'csv',
'player_transfer_fee.csv')}' with {len(transfer_data)} records")
else:
    print("No matching players found.")
```

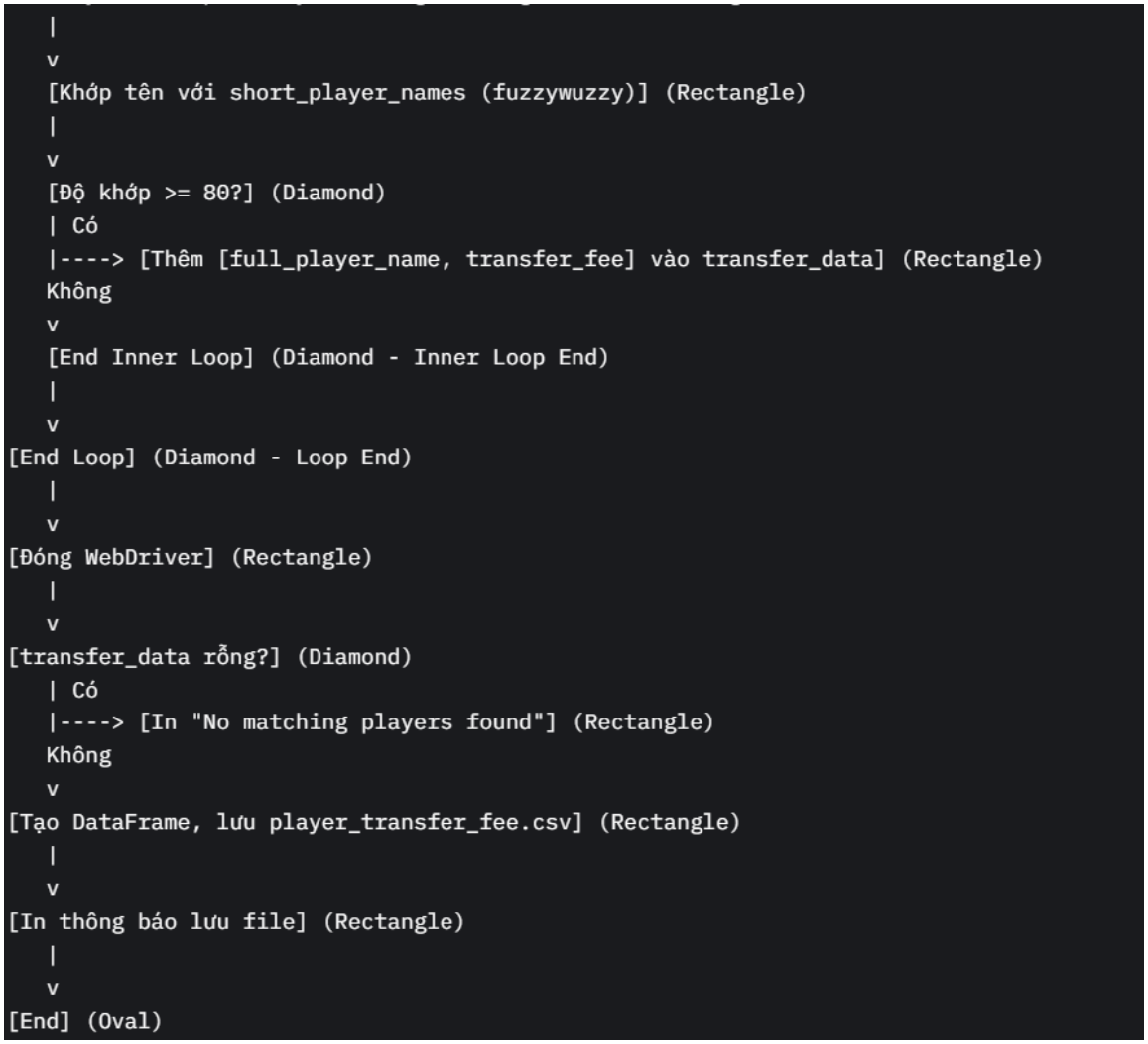
- **Cách hoạt động:**
  + Kiểm tra dữ liệu:
    - Kiểm tra xem transfer_data có dữ liệu không (if transfer_data).
  + Tạo DataFrame:

- pd.DataFrame(transfer_data, columns=['Player', 'Price']) tạo DataFrame từ transfer_data với 2 cột: Player (tên cầu thủ) và Price (phí chuyển nhượng).

+ Lưu file:
  - Lưu vào player_transfer_fee.csv bằng to_csv, không lưu chỉ số (index=False).
  - In đường dẫn file và số lượng bản ghi.
+ Trường hợp không có dữ liệu:
  - Nếu transfer_data rỗng, in thông báo "No matching players found."

3.1.7 Sơ đồ luồng tổng quát


[Start] (Oval)
|
v
[Thiết lập root_dir, csv_folder, result_csv_path] (Rectangle)
|
v
[Kiểm tra result.csv tồn tại?] (Diamond)
| Không
|----> [In lỗi, thoát] (Rectangle)
Có
v
[Đọc result.csv vào data_frame] (Rectangle)
|
v
[Lọc Minutes > 900, lưu players_over_900_minutes.csv] (Rectangle)
|
v
[Đọc players_over_900_minutes.csv, rút ngắn tên cầu thủ] (Rectangle)
|
v
[Cấu hình Selenium: headless, user-agent] (Rectangle)
|
v
[Khởi tạo WebDriver, tạo transfer_urls (trang 1-14)] (Rectangle)
|
v
[For url in transfer_urls] (Diamond - Loop Start)
|
v
[Truy cập url, chờ transfer-table] (Rectangle)
|
v
[Lấy rows từ bảng] (Rectangle)
|
v
[For row in rows] (Diamond - Inner Loop Start)
|
v
[Lấy tên và phí chuyển nhượng, rút ngắn tên] (Rectangle)



3.2 Ước tính giá trị cầu thủ

3.2.1 Chuẩn bị dữ liệu huấn luyện mô hình

- Thu thập dữ liệu:
    + Code sử dụng Selenium để cào dữ liệu ETV của các cầu thủ ở giải Ngoại hạng Anh từ trang FootballTransfers.
        - Tên cầu thủ: Định danh cho từng cầu thủ.
        - Vị trí: Đặc trưng phân loại (ví dụ: GK - thủ môn, DF - hậu vệ, MF - tiền vệ, FW - tiền đạo), rất quan trọng vì giá trị chuyển nhượng khác nhau theo vị trí.
        - Giá trị ETV (Price): Biến mục tiêu (giá trị chuyển nhượng ước tính), dùng làm giá trị thực để huấn luyện mô hình.
    + Code so khớp tên cầu thủ với tệp result.csv (nếu có) bằng cách sử dụng thuật toán fuzzy matching (fuzzywuzzy), đảm bảo tính chính xác và thống nhất của dữ liệu.

- Tổ chức dữ liệu:
    + Code phân loại cầu thủ theo vị trí (GK, DF, MF, FW), giúp phân tích và xây dựng mô hình riêng cho từng vị trí, vì giá trị chuyển nhượng thay đổi theo vai trò.
    + Loại bỏ dữ liệu trùng lặp và lưu vào tệp CSV (all_estimate_transfer_fee.csv), sẵn sàng cho phân tích hoặc huấn luyện mô hình.
- Xử lý thách thức:
    + Code sử dụng Selenium để xử lý nội dung động và phân trang, thu thập tối đa dữ liệu từ các trang (lên đến 22 trang hoặc số trang phát hiện được).
    + Xử lý lỗi (ví dụ: thử lại 3 lần nếu gặp lỗi timeout, đặt số trang mặc định nếu không phát hiện phân trang) để đảm bảo thu thập dữ liệu đáng tin cậy.
    + Fuzzy matching với ngưỡng 70% giúp khớp tên cầu thủ chính xác, giảm thiểu mất mát dữ liệu do khác biệt về cách viết tên.
- Nền tảng cho mô hình hóa:
    + Tệp all_estimate_transfer_fee.csv cung cấp biến mục tiêu (ETV), cần thiết để huấn luyện mô hình học máy có giám sát.
    + Đặc trưng vị trí có thể được sử dụng trực tiếp hoặc kết hợp với các đặc trưng bổ sung (như tuổi, số bàn thắng) để xây dựng mô hình định giá toàn diện.

3.2.2 Thiết lập môi trường và cấu hình

- **Code:**

```
base_dir = r"C:\Users\DD\OneDrive\Documents\newfolder(2)\btlpython"
csv_dir = os.path.join(base_dir, "csv")
result_path = os.path.join(csv_dir, "result.csv")
etv_path = os.path.join(csv_dir, 'all_estimate_transfer_fee.csv')

standard_output_columns = [
    'Player', 'Team', 'Nation', 'Position', 'Actual_Transfer_Value_M',
'Predicted_Transfer_Value_M'
]

roles_config = {
    'Goalkeeper': {
        'role_filter': 'GK',
        'data_path': etv_path,
        'attributes': [
            'Save%', 'CS%', 'GA90', 'Minutes', 'Age', 'PK Save%', 'Team', 'Nation'
        ],
        'key_attributes': ['Save%', 'CS%', 'PK Save%']
    },
    'Defender': {...},
    'Midfielder': {...},
    'Forward': {...}
}
```

- **Cách hoạt động:**
  + Thư mục và file:
    - base_dir: Thư mục gốc chứa dự án.
    - csv_dir: Thư mục con csv chứa các file CSV.
    - result_path: Đường dẫn đến result.csv (dữ liệu thống kê cầu thủ).
    - etv_path: Đường dẫn đến all_estimate_transfer_fee.csv (dữ liệu giá trị chuyển nhượng).
  + Cột đầu ra:
    - standard_output_columns: Định nghĩa các cột trong file đầu ra: Tên cầu thủ, Đội, Quốc tịch, Vị trí, Giá trị thực tế (triệu Euro), Giá trị dự đoán (triệu Euro).
  1. Cấu hình vị trí:
     - roles_config: Từ điển định nghĩa cho 4 vị trí (Thủ môn, Hậu vệ, Tiền vệ, Tiền đạo), mỗi vị trí có:
       - role_filter: Mã vị trí (GK, DF, MF, FW) để lọc dữ liệu.
       - data_path: File chứa giá trị chuyển nhượng (etv_path).
       - attributes: Danh sách đặc trưng dùng để dự đoán (ví dụ: Thủ môn dùng Save%, CS%; Tiền đạo dùng Gls, xG per 90).
       - key_attributes: Các đặc trưng quan trọng được tăng trọng số sau này.

3.2.3 Hàm phụ trợ

- **Code:**

```python
def simplify_name(name):
    if not isinstance(name, str):
        return ""
    parts = name.strip().split()
    return " ".join(parts[:2]) if len(parts) >= 2 else name

def convert_valuation(val_text):
    if pd.isna(val_text) or val_text in ["N/A", ""]:
        return np.nan
    try:
        val_text = re.sub(r'[€£]', '', val_text).strip().upper()
        multiplier = 1000000 if 'M' in val_text else 1000 if 'K' in val_text else 1
        value = float(re.sub(r'[MK]', '', val_text)) * multiplier
        return value
    except (ValueError, TypeError):
        return np.nan

def match_player_name(name, options, min_score=90):
    if not isinstance(name, str):
        return None, None
    simplified_name = simplify_name(name).lower()
    simplified_options = [simplify_name(opt).lower() for opt in options if
isinstance(opt, str)]
    match = process.extractOne(
        simplified_name,
        simplified_options,
        scorer=fuzz.token_sort_ratio,
        score_cutoff=min_score
    )
    if match is not None:
        matched_idx = simplified_options.index(match[0])
        return options[matched_idx], match[1]
    return None, None
```

- **Cách hoạt động:**
  + Hàm simplify_name:
    - Rút ngắn tên cầu thủ bằng cách lấy 2 phần đầu tiên (ví dụ: "Mohamed Salah Eldin" → "Mohamed Salah").
    - Xử lý trường hợp không phải chuỗi (NaN, số) bằng cách trả về chuỗi rỗng.
    - Mục đích: Chuẩn hóa tên để khớp giữa hai file dữ liệu.
  + Hàm convert_valuation:
    - Chuyển đổi giá trị chuyển nhượng (chuỗi như "€50M", "£20K") thành số (đơn vị Euro):
      - Loại bỏ ký hiệu tiền tệ (€, £) bằng re.sub.
      - Xác định đơn vị: M (triệu) → nhân 1,000,000; K (nghìn) → nhân 1,000.
      - Chuyển thành float, trả về NaN nếu lỗi.
    - Mục đích: Chuẩn hóa giá trị chuyển nhượng để sử dụng trong mô hình.
  + Hàm match_player_name:
    - Sử dụng fuzzywuzzy để khớp tên cầu thủ từ result.csv với tên trong all_estimate_transfer_fee.csv.

- Rút ngắn tên bằng simplify_name, chuyển thành chữ thường để so sánh.
- fuzz.token_sort_ratio: So sánh tên dựa trên từ, bỏ qua thứ tự (ví dụ: "Salah Mohamed" khớp với "Mohamed Salah").
- Chỉ chấp nhận khớp nếu điểm số ≥ min_score=90 (ngưỡng cao để đảm bảo chính xác).
- Trả về tên khớp và điểm số, hoặc (None, None) nếu không khớp.

3.2.3 Hàm xử lý dữ liệu theo vị trí (analyze_role)

- **Code:**

```
def analyze_role(role, config):
    # Đọc dữ liệu
    try:
        stats_data = pd.read_csv(result_path)
        valuation_data = pd.read_csv(config['data_path'])
    except FileNotFoundError as e:
        print(f"Lỗi: Không tìm thấy tệp cho {role} - {e}")
        return None, None

    # Lọc dữ liệu theo vị trí
    stats_data['Main_Role'] = stats_data['Position'].astype(str).str.split(r'[,/]').str[
    stats_data = stats_data[stats_data['Main_Role'].str.upper() == config['role_filter']

    # Khớp tên cầu thủ
    player_list = valuation_data['Player'].dropna().tolist()
    stats_data['Linked_Name'] = None
    stats_data['Link_Score'] = None
    stats_data['Valuation'] = np.nan

    for idx, row in stats_data.iterrows():
        linked_name, score = match_player_name(row['Player'], player_list)
        if linked_name:
            stats_data.at[idx, 'Linked_Name'] = linked_name
            stats_data.at[idx, 'Link_Score'] = score
            linked_row = valuation_data[valuation_data['Player'] == linked_name]
            if not linked_row.empty:
                val_value = convert_valuation(linked_row['Price'].iloc[0])
                stats_data.at[idx, 'Valuation'] = val_value
```

```
# Lọc dữ liệu đã khớp
filtered_data = stats_data[stats_data['Linked_Name'].notna()].copy()
filtered_data = filtered_data.drop_duplicates(subset='Linked_Name')

# Báo cáo cầu thủ không khớp
unmatched_players = stats_data[stats_data['Linked_Name'].isna()]['Player'].dropna().
if unmatched_players:
    print(f"Cầu thủ {role} không khớp: {len(unmatched_players)} cầu thủ không được k
    print(unmatched_players)

# Xử lý dữ liệu đặc trưng
attributes = config['attributes']
target_col = 'Valuation'
for col in attributes:
    if col in ['Team', 'Nation']:
        filtered_data[col] = filtered_data[col].fillna('Unknown')
    else:
        filtered_data[col] = pd.to_numeric(filtered_data[col], errors='coerce')
        median_val = filtered_data[col].median()
        filtered_data[col] = filtered_data[col].fillna(median_val if not pd.isna(med

# Chuẩn hóa dữ liệu số
numeric_attrs = [col for col in attributes if col not in ['Team', 'Nation']]
for col in numeric_attrs:
    filtered_data[col] = np.log1p(filtered_data[col].clip(lower=0))
```

```
# Tăng trọng số đặc trưng
for col in config['key_attributes']:
    if col in filtered_data.columns:
        filtered_data[col] = filtered_data[col] * 2.0
if 'Minutes' in filtered_data.columns:
    filtered_data['Minutes'] = filtered_data['Minutes'] * 1.5
if 'Age' in filtered_data.columns:
    filtered_data['Age'] = filtered_data['Age'] * 0.5

# Chuẩn bị dữ liệu học máy
ml_data = filtered_data.dropna(subset=[target_col]).copy()
if ml_data.empty:
    print(f"Lỗi: Không có dữ liệu Valuation hợp lệ cho {role}.")
    return None, unmatched_players

X = ml_data[attributes]
y = ml_data[target_col]

# Chia tập huấn luyện/kiểm tra
if len(ml_data) > 5:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
else:
    print(f"Cảnh báo: Không đủ dữ liệu cho {role} để chia tập huấn luyện/kiểm tra.")
    X_train, y_train = X, y
    X_test, y_test = X, y

# Tạo pipeline
categorical_attrs = [col for col in attributes if col in ['Team', 'Nation']]
data_transformer = ColumnTransformer(
```

```
        transformers=[
            ('num', StandardScaler(), numeric_attrs),
            ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), categor
        ])

    model_pipeline = Pipeline([
        ('transformer', data_transformer),
        ('model', LinearRegression())
    ])

    # Huấn luyện mô hình
    model_pipeline.fit(X_train, y_train)

    # Đánh giá mô hình
    if len(X_test) > 0:
        y_pred = model_pipeline.predict(X_test)
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))
        r2 = r2_score(y_test, y_pred)

    # Dự đoán giá trị
    filtered_data['Estimated_Value'] = model_pipeline.predict(filtered_data[attributes])
    filtered_data['Estimated_Value'] = filtered_data['Estimated_Value'].clip(lower=100_0
    filtered_data['Predicted_Transfer_Value_M'] = (filtered_data['Estimated_Value'] / 1_
    filtered_data['Actual_Transfer_Value_M'] = (filtered_data['Valuation'] / 1_000_000).

    # Chuẩn bị đầu ra
    for col in standard_output_columns:
        if col not in filtered_data.columns:
            filtered_data[col] = np.nan if col in ['Actual_Transfer_Value_M', 'Predicted
    filtered_data['Position'] = role
    output_data = filtered_data[standard_output_columns].copy()

    # Khôi phục dữ liệu gốc
    numeric_attrs_no_age = [col for col in numeric_attrs if col != 'Age']
    for col in numeric_attrs_no_age:
        if col in output_data.columns:
            output_data[col] = np.expm1(output_data[col]).round(2)
    if 'Age' in output_data.columns:
        output_data['Age'] = np.expm1(output_data['Age']).round(0)
        median_age = output_data['Age'].median()
        output_data['Age'] = output_data['Age'].fillna(median_age).astype(int)

    return output_data, unmatched_players
```

- **Cách hoạt động:**
  + Đọc dữ liệu:
    - Đọc result.csv (thống kê) và all_estimate_transfer_fee.csv (giá trị chuyển nhượng).
    - Xử lý lỗi file không tồn tại bằng try-except.
  + Lọc theo vị trí:
    - Tạo cột Main_Role bằng cách lấy vị trí đầu tiên từ cột Position (ví dụ: "DF,MF" → "DF").
    - Lọc dữ liệu theo role_filter (ví dụ: chỉ giữ cầu thủ có Main_Role == 'GK' cho thủ môn).
  + Khớp tên cầu thủ:

- Lặp qua từng cầu thủ trong stats_data, sử dụng match_player_name để khớp với tên trong valuation_data.
- Lưu tên khớp (Linked_Name), điểm số (Link_Score), và giá trị chuyển nhượng (Valuation).

+ Lọc dữ liệu đã khớp:
  - Giữ các cầu thủ có Linked_Name không rỗng, loại bỏ trùng lặp dựa trên Linked_Name.
  - Báo cáo danh sách cầu thủ không khớp (unmatched_players).
+ Xử lý đặc trưng:
  - Danh mục: Điền Unknown cho Team, Nation nếu thiếu.
  - Số: Chuyển thành số, điền NaN bằng trung vị cột (hoặc 0 nếu trung vị thiếu).
  - Chuẩn hóa: Áp dụng log1p (log(1+x)) cho các đặc trưng số để giảm độ lệch, giới hạn giá trị nhỏ nhất là 0.
  - Tăng trọng số:
    - Nhân đôi giá trị các key_attributes (ví dụ: Save% cho thủ môn).
    - Nhân Minutes với 1.5, Age với 0.5 để điều chỉnh tầm quan trọng.
+ Chuẩn bị dữ liệu học máy:
  - Loại bỏ hàng thiếu Valuation (biến mục tiêu).
  - Tách đặc trưng (X: các attributes) và mục tiêu (y: Valuation).
+ Chia tập dữ liệu:
  - Nếu có >5 mẫu, chia 80% huấn luyện, 20% kiểm tra (train_test_split).
  - Nếu ≤5 mẫu, dùng toàn bộ dữ liệu để huấn luyện và kiểm tra.
+ Tạo pipeline:
  - ColumnTransformer:
    - Chuẩn hóa đặc trưng số bằng StandardScaler.
    - Mã hóa đặc trưng danh mục (Team, Nation) bằng OneHotEncoder.
  - Pipeline: Kết hợp transformer và mô hình LinearRegression.
+ Huấn luyện và đánh giá:
  - Huấn luyện mô hình trên X_train, y_train.
  - Đánh giá trên X_test bằng RMSE (Root Mean Squared Error) và R^2 (độ giải thích).
+ Dự đoán giá trị:
  - Dự đoán giá trị chuyển nhượng cho toàn bộ filtered_data.
  - Giới hạn giá trị trong khoảng [100,000, 200,000,000] Euro.

- Chuyển thành triệu Euro (Predicted_Transfer_Value_M, Actual_Transfer_Value_M).

+ Chuẩn bị đầu ra:
  - Đảm bảo các cột trong standard_output_columns.
  - Gán Position là tên vai trò (ví dụ: "Goalkeeper").
  - Khôi phục dữ liệu gốc bằng expm1 (ngược của log1p) cho đặc trưng số, trừ Age.
  - Xử lý Age riêng, điền trung vị nếu thiếu.
+ Trả về:
  - DataFrame output_data với các cột đầu ra.
  - Danh sách unmatched_players.

3.2.4 Tổng hợp và lưu kết quả

- **Code:**

```
combined_outputs = []
unmatched_records = []

for role, config in roles_config.items():
    print(f"\nĐang xử lý {role}...")
    output, unmatched = analyze_role(role, config)
    if output is not None:
        combined_outputs.append(output)
    if unmatched:
        unmatched_records.extend([(role, player) for player in unmatched])

if combined_outputs:
    final_output = pd.concat(combined_outputs, ignore_index=True)
    final_output = final_output.sort_values(by='Predicted_Transfer_Value_M',
ascending=False)
    final_output.to_csv(os.path.join(csv_dir, 'ml_estimated_values_linear.csv'),
index=False)
    print(f"Giá trị ước tính của các cầu thủ đã được lưu vào '{os.path.join(csv_dir,
'ml_estimated_values_linear.csv')}'")
```

- **Cách hoạt động:**

+ Lặp qua các vị trí:
  - Gọi analyze_role cho từng vị trí trong roles_config.
  - Lưu output (DataFrame đầu ra) vào combined_outputs.
  - Lưu unmatched_players vào unmatched_records.
+ Tổng hợp đầu ra:
  - Nếu combined_outputs không rỗng, gộp các DataFrame bằng pd.concat.
  - Sắp xếp theo Predicted_Transfer_Value_M (giảm dần) để hiển thị cầu thủ có giá trị cao nhất trước.
+ Lưu file:
  - Lưu vào ml_estimated_values_linear.csv với các cột standard_output_columns.

- In thông báo xác nhận.

3.2.5 Nhận xét về lựa chọn đặc trưng và mô hình

3.2.5.1 Chọn đặc trưng

- **Tiêu chí chọn đặc trưng**
  + Liên quan đến mục tiêu: Đặc trưng phải có mối quan hệ với biến mục tiêu (giá trị chuyển nhượng). Ví dụ, số bàn thắng (Gls) ảnh hưởng mạnh đến giá trị của tiền đạo.
  + Đại diện cho vai trò: Các đặc trưng nên phản ánh đặc điểm của từng vị trí cầu thủ (thủ môn, hậu vệ, tiền vệ, tiền đạo).
  + Độ biến thiên: Đặc trưng có giá trị đa dạng (không quá đồng nhất) để cung cấp thông tin hữu ích.
  + Không dư thừa: Tránh các đặc trưng có tương quan cao (multicollinearity) để giảm phức tạp mô hình.
  + Dễ xử lý: Đặc trưng nên dễ chuẩn hóa, ít giá trị thiếu, hoặc dễ điền giá trị thiếu.
- **Phương pháp chọn đặc trưng**
  + Dựa trên kiến thức chuyên môn (Domain Knowledge):
    - Hiểu ngữ cảnh bóng đá để chọn các chỉ số thống kê phù hợp với từng vị trí.
    - Ví dụ: Thủ môn cần Save% (tỷ lệ cứu thua), Tiền đạo cần Gls per 90 (bàn thắng mỗi 90 phút).
  + Phân tích dữ liệu (Exploratory Data Analysis - EDA):
    - Kiểm tra tương quan giữa đặc trưng và mục tiêu (bằng Pearson correlation).
    - Loại bỏ đặc trưng có giá trị thiếu quá nhiều hoặc không biến thiên.
  + Kỹ thuật tự động:
    - Feature Importance: Sử dụng mô hình như Random Forest để xếp hạng đặc trưng.
    - Recursive Feature Elimination (RFE): Loại bỏ dần đặc trưng không quan trọng.
    - Regularization: Sử dụng L1 (Lasso) để tự động chọn đặc trưng.
  + Thử nghiệm: Thử các tập hợp đặc trưng khác nhau, đánh giá hiệu suất mô hình (RMSE, R^2).
- **Cách đoạn code chọn đặc trưng**
  + Trong đoạn code, đặc trưng được chọn thông qua cấu hình roles_config, dựa trên kiến thức chuyên môn bóng đá và vai trò cầu thủ:
    - Cấu hình roles_config:
      - Mỗi vị trí (Thủ môn, Hậu vệ, Tiền vệ, Tiền đạo) có danh sách attributes và key_attributes:

        - Thủ môn: Save%, CS% (tỷ lệ giữ sạch lưới), GA90 (bàn thua mỗi 90 phút), Minutes, Age, PK Save%, Team, Nation.
        - Hậu vệ: Tkl (tắc bóng), Int (đánh chặn), Blocks, Recov (thu hồi bóng), Aerl Won% (thắng tranh chấp trên không), v.v.
        - Tiền vệ: KP (đường chuyền then chốt), PPA (chuyền vào khu vực nguy hiểm), PrgP (chuyền tiến bộ), SCA (hành động tạo bàn), v.v.
        - Tiền đạo: Gls, Ast, Gls per 90, xG per 90, SCA90, GCA90, v.v.
    - Key_attributes: Các đặc trưng quan trọng được nhân trọng số 2.0 để tăng tầm quan trọng (ví dụ: Save% cho thủ môn, Gls cho tiền đạo).
+ Xử lý đặc trưng:
    - Số: Chuyển thành số, điền NaN bằng trung vị, áp dụng log1p để giảm độ lệch (skewness).
    - Danh mục: Team, Nation được mã hóa bằng OneHotEncoder, điền Unknown nếu thiếu.
    - Tăng trọng số:
        - Minutes nhân 1.5 (phản ánh thời gian thi đấu quan trọng).
        - Age nhân 0.5 (tuổi ít quan trọng hơn).
+ Lý do chọn:
    - Các đặc trưng được chọn dựa trên vai trò cầu thủ:
        - Thủ môn: Tập trung vào khả năng phòng ngự (cứu thua, giữ sạch lưới).
        - Hậu vệ: Nhấn mạnh tắc bóng, đánh chặn, tranh chấp.
        - Tiền vệ: Tạo cơ hội (chuyền, kiến tạo).
        - Tiền đạo: Ghi bàn và tạo bàn.
        - Bao gồm cả đặc trưng số (hiệu suất) và danh mục (Team, Nation) để mô hình học được các yếu tố bổ sung (ví dụ: đội mạnh tăng giá trị cầu thủ).
        - Key_attributes được ưu tiên vì chúng đặc trưng nhất cho hiệu suất (ví dụ: Gls quyết định giá trị tiền đạo).

3.2.5.2 Chọn mô hình

- Tiêu chí chọn mô hình
    + Loại bài toán: Đây là bài toán hồi quy (dự đoán giá trị liên tục: giá trị chuyển nhượng).
    + Độ phức tạp của dữ liệu:
        - Dữ liệu tuyến tính → Linear Regression, Ridge, Lasso.

        - Dữ liệu phi tuyến → Random Forest, Gradient Boosting (XGBoost, LightGBM).
    + Kích thước dữ liệu:
        - Dữ liệu nhỏ → Mô hình đơn giản (Linear Regression).
        - Dữ liệu lớn → Mô hình phức tạp hơn (Deep Learning, Ensemble).
    + Khả năng diễn giải: Mô hình đơn giản (Linear Regression) dễ diễn giải hơn mô hình phức tạp (Neural Networks).
    + Hiệu suất: Đánh giá bằng các chỉ số như RMSE (lỗi bình phương trung bình gốc) và R^2 (độ giải thích).
    + Tốc độ huấn luyện: Mô hình nhanh hơn (Linear Regression) phù hợp với dữ liệu nhỏ hoặc cần xử lý nhanh.
- Phương pháp chọn mô hình
    + Bắt đầu với mô hình đơn giản:
        - Thử Linear Regression để kiểm tra giả định tuyến tính.
        - Nếu R^2 thấp hoặc RMSE cao, chuyển sang mô hình phức tạp hơn.
    + So sánh nhiều mô hình:
        - Thử các mô hình hồi quy: Linear Regression, Ridge, Random Forest, XGBoost.
        - Sử dụng cross-validation để đánh giá hiệu suất:

```
from sklearn.model_selection import cross_val_score
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(random_state=42),
    'XGBoost': XGBRegressor(random_state=42)
}
for name, model in models.items():
    scores = cross_val_score(model, X_transformed, y, cv=5, scoring='r2')
    print(f"{name}: R2 = {scores.mean():.2f} ± {scores.std():.2f}")
```

- Điều chỉnh siêu tham số (Hyperparameter Tuning):
    + Sử dụng GridSearchCV hoặc RandomizedSearchCV để tìm tham số tối ưu (ví dụ: số cây trong Random Forest).
- Đánh giá trên tập kiểm tra:
    + So sánh RMSE, R^2 trên tập kiểm tra để chọn mô hình tốt nhất.
- Xem xét yêu cầu thực tiễn:
    + Nếu cần diễn giải (ví dụ: hiểu đặc trưng nào quan trọng), chọn Linear Regression.
    + Nếu cần độ chính xác cao, chọn Random Forest hoặc XGBoost.
- Cách đoạn code chọn mô hình
- Trong đoạn code, mô hình được chọn là Linear Regression, tích hợp trong một pipeline:
    + Pipeline:

```
model_pipeline = Pipeline([
    ('transformer', ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numeric_attrs),
            ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False),
categorical_attrs)
        ])),
    ('model', LinearRegression())
])
```

ColumnTransformer: Chuẩn hóa đặc trưng số (StandardScaler) và mã hóa đặc trưng danh mục (OneHotEncoder).

+ LinearRegression: Mô hình hồi quy tuyến tính dự đoán Valuation (giá trị chuyển nhượng).

- Huấn luyện và đánh giá:
  + Huấn luyện trên tập huấn luyện (X_train, y_train).
  + Đánh giá trên tập kiểm tra bằng RMSE và R^2 (nếu có đủ dữ liệu):

```
y_pred = model_pipeline.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

- Dự đoán:
  + Dự đoán giá trị chuyển nhượng cho toàn bộ dữ liệu (filtered_data), giới hạn trong khoảng [100,000, 200,000,000] Euro.
- Lý do chọn Linear Regression:
  + Đơn giản: Linear Regression dễ triển khai, nhanh, và phù hợp với dữ liệu nhỏ (code kiểm tra len(ml_data) > 5).
  + Diễn giải được: Hệ số của mô hình cho biết mức độ ảnh hưởng của mỗi đặc trưng (ví dụ: Gls tăng 1 đơn vị làm tăng bao nhiêu giá trị chuyển nhượng).
  + Giả định tuyến tính: Code giả định mối quan hệ giữa đặc trưng (như Gls, Save%) và giá trị chuyển nhượng là tuyến tính, đặc biệt sau khi áp dụng log1p để giảm độ lệch.
  + Dữ liệu danh mục: OneHotEncoder cho Team, Nation phù hợp với Linear Regression, vì nó tạo ra các biến giả (dummy variables).