HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH NGÔN NGỮ PYTHON

BÁO CÁO VỀ BÀI TẬP LỚN

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**

Lớp : **D23CQCE06-B**

Họ Tên : **LÊ ĐỨC MẠNH**

MSV : **B23DCCE063**

Hà Nội _ 2025

LỜI MỞ ĐẦU

Trong thời đại dữ liệu phát triển mạnh mẽ như hiện nay, việc thu thập, xử lý và phân tích dữ liệu đóng vai trò then chốt trong nhiều lĩnh vực, đặc biệt là trong thể thao, nơi mà mỗi con số đều có thể tác động đến chiến thuật, giá trị chuyển nhượng hay thành tích của đội bóng. Với mong muốn vận dụng những kiến thức đã học về lập trình Python vào thực tiễn, em đã thực hiện bài tập lớn với chủ đề **"Phân tích dữ liệu thống kê cầu thủ Premier League mùa giải 2024–2025"**.

Bài báo cáo này trình bày quá trình xây dựng một hệ thống thu thập và phân tích dữ liệu cầu thủ từ trang FBref.com — một trang web cung cấp số liệu thống kê bóng đá uy tín. Dữ liệu thu thập được sẽ được xử lý, phân tích thống kê cơ bản, trực quan hóa bằng biểu đồ và ứng dụng một số phương pháp học máy để dự đoán giá trị chuyển nhượng cầu thủ.

Cụ thể, bài tập được chia thành 4 bài chính:

Bài 1: Tự động thu thập dữ liệu thống kê cầu thủ Premier League có thời gian thi đấu trên 90 phút bằng Selenium và BeautifulSoup.

Bài 2: Phân tích Top 3 cầu thủ có chỉ số cao nhất và thấp nhất ở từng hạng mục, tính toán thống kê mô tả và trực quan hóa phân phối dữ liệu bằng biểu đồ.

Bài 3: Phân tích toàn bộ các chỉ số dạng số, xác định các chỉ số hợp lệ, tính toán thống kê và đánh giá đội bóng có phong độ tốt nhất dựa trên các chỉ số tấn công.

Bài 4: Thu thập dữ liệu giá trị chuyển nhượng từ các trang chuyên về bóng đá, chuẩn hóa dữ liệu và ứng dụng mô hình Random Forest để dự đoán giá trị chuyển nhượng của các cầu thủ dựa trên các chỉ số thống kê chuyên môn.

Thông qua bài tập này, em có cơ hội vận dụng kiến thức về **web scraping**, **pandas**, **matplotlib**, **machine learning** và các kỹ năng lập trình xử lý dữ liệu thực tế. Đồng thời, bài tập cũng giúp em hiểu rõ hơn về cách tổ chức và phân tích dữ liệu thể thao một cách khoa học, hỗ trợ cho việc ra quyết định trong bóng đá hiện đại.

Em xin chân thành cảm ơn thầy **Kim Ngọc Bách** đã tận tình hướng dẫn và tạo điều kiện để em thực hiện bài tập này.

Bài 1: Lấy dữ liệu cầu thủ từ trang FBref

1:Import các thư viện chính

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

```
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 os
import uuid
import time
```

- **Ý nghĩa**: Nhập các thư viện cần thiết để:
  - Quản lý thời gian (time), xử lý dữ liệu bảng (pandas), phân tích HTML (BeautifulSoup).
  - Tự động hóa trình duyệt (selenium, webdriver_manager), xử lý chuỗi HTML (StringIO), và quản lý file/thư mục (os).
  - Lưu ý: uuid không được sử dụng (có thể dư thừa).
- **Mục đích**: Chuẩn bị công cụ cho việc crawl, xử lý, và lưu dữ liệu.

2. Thiết lập thư mục và cấu hình Selenium

```
# Thiết lập thư mục lưu trữ file kết quả
output_dir = os.path.join(os.getcwd(), "output")
# Cấu hình Selenium
options = Options()
options.add_argument("--headless")
options.add_argument("--disable-gpu")
options.add_argument("--no-sandbox")
# Khởi tạo WebDriver
try:
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
except Exception as e:
    print(f"Lỗi khi khởi tạo WebDriver: {e}")
    exit(1)
```

Ý nghĩa:

Tạo thư mục output để lưu file kết quả (results.csv).

Cấu hình Selenium chạy Chrome ở chế độ ẩn (--headless), tắt GPU (--disable-gpu), và bỏ sandbox (--no-sandbox) để tăng hiệu suất.

Khởi tạo ChromeDriver, xử lý lỗi nếu thất bại.

Mục đích: Thiết lập môi trường lưu trữ và công cụ tự động hóa trình duyệt.

3. Hàm hỗ trợ xử lý dữ liệu

```
# Hàm chuyển tuổi từ định dạng "năm-ngày" sang số thập phân
```

```
def parse_age(age_input):
    try:
        age_str = str(age_input).replace(" ", "")
        if "-" in age_str:
            years, days = age_str.split("-")
            return round(int(years) + (int(days) / 365), 2)
        return "N/A"
    except (ValueError, AttributeError):
        return "N/A"


# Hàm trích xuất mã quốc tịch
def extract_country_code(nation_str):
    try:
        return nation_str.split()[-1]
    except (AttributeError, IndexError):
        return "N/A"
```

- **Ý nghĩa**:
  - parse_age: Chuyển tuổi từ "năm-ngày" (VD: "25-100") thành số thập phân (VD: 25.27).
  - extract_country_code: Lấy mã quốc tịch (VD: "England ENG" -> "ENG").
  - Trả về "N/A" nếu dữ liệu không hợp lệ.
- **Mục đích**: Chuẩn hóa dữ liệu tuổi và quốc tịch để đồng nhất và dễ xử lý.

4. Cấu hình nguồn dữ liệu và cột yêu cầu

```
# Cấu hình nguồn dữ liệu
urls = [
    "https://fbref.com/en/comps/9/2024-2025/stats/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/keepers/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/shooting/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/passing/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/gca/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/defense/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/possession/2024-2025-Premier-League-Stats",
    "https://fbref.com/en/comps/9/2024-2025/misc/2024-2025-Premier-League-Stats",
]

table_ids = [
    "stats_standard",
```

```
    "stats_keeper",
    "stats_shooting",
    "stats_passing",
    "stats_gca",
    "stats_defense",
    "stats_possession",
    "stats_misc",
]
# Danh sách cột yêu cầu
required_columns = [
    "Player", "Nation", "Team", "Position", "Age",
    "Matches Played", "Starts", "Minutes",
    "Gls", "Ast", "xG", "xAG", "Gls per 90", "Ast per 90", "xG per 90", "xAG per 90",
    "SoT%", "SoT per 90", "G per Sh", "Dist",
    "GA90", "Save%", "CS%", "PK Save%",
    "Cmp", "Cmp%", "TotDist", "ShortCmp%", "MedCmp%", "LongCmp%", "KP", "Pass into 1_3", "PPA", "CrsPA",
    "SCA", "SCA90", "GCA", "GCA90",
    "Tkl", "TklW", "Deff Att", "Lost", "Blocks", "Sh", "Pass", "Int",
    "Touches", "Def Pen", "Def 3rd", "Mid 3rd", "Att 3rd", "Att Pen", "Take-Ons Att", "Succ%", "Tkld%",
    "Carries", "ProDist", "Carries 1_3", "CPA", "Mis", "Dis", "Rec", "Rec PrgR",
    "Fls", "Fld", "Off", "Crs", "Recov", "Aerl Won", "Aerl Lost", "Aerl Won%"
]
```

- **Ý nghĩa**:
  - urls: Danh sách các trang web từ fbref.com chứa dữ liệu thống kê (tổng quan, thủ môn, sút bóng, chuyền bóng, v.v.).
  - table_ids: ID của các bảng HTML tương ứng trên mỗi trang.
  - required_columns: Danh sách các cột mong muốn trong file kết quả, bao gồm thông tin cá nhân và thống kê (bàn thắng, kiến tạo, phòng ngự, v.v.).
- **Mục đích**: Xác định nguồn dữ liệu và cấu trúc đầu ra mong muốn.

5. Từ điển đổi tên cột

```
# Từ điển đổi tên cột
column_rename_dict = {
    "stats_shooting": {
        "Standard.8": "Dist",
        "Standard.6": "G per Sh",
        "Standard.5": "SoT per 90",
        "Standard.3": "SoT%",
        "Unnamed: 4": "Team",
        "Unnamed: 1": "Player",
```

```
    },
    "stats_keeper": {
        "Penalty Kicks.4": "PK Save%",
        "Performance.9": "CS%",
        "Performance.4": "Save%",
        "Performance.1": "GA90",
        "Unnamed: 4": "Team",
        "Unnamed: 1": "Player",
    },
    # ... (tương tự cho các bảng khác)
}
```

- **Ý nghĩa**:
  - Ánh xạ tên cột gốc từ HTML (VD: "Standard.8", "Unnamed: 1") sang tên chuẩn hóa (VD: "Dist", "Player").
  - Mỗi bảng (stats_shooting, stats_keeper, v.v.) có từ điển riêng.
- **Mục đích**: Chuẩn hóa tên cột để đồng nhất dữ liệu từ các bảng, dễ gộp và xử lý sau này.

6. Trích xuất dữ liệu từ trang web

```
# Trích xuất dữ liệu
all_tables = {}
for url, table_id in zip(urls, table_ids):
    print(f"Processing {table_id} from {url}")
    driver.get(url)
    time.sleep(2)

    soup = BeautifulSoup(driver.page_source, "html.parser")
    table = soup.find("table", {"id": table_id})

    if not table:
        print(f"Table {table_id} not found!")
        continue

    df = pd.read_html(StringIO(str(table)), header=0)[0]
    print(f"Original columns in {table_id}:", df.columns.tolist())

    # Kiểm tra cột Player và Team trước
    df = df.rename(columns=column_rename_dict.get(table_id))
    if "Player" not in df.columns or "Team" not in df.columns:
        print(f"Bỏ qua {table_id}: Thiếu cột 'Player' hoặc 'Team'")
        continue

    # Loại bỏ cột trùng lặp
    df = df.loc[:, ~df.columns.duplicated()]
```

```
    # Lọc cột cần thiết
    required_for_table = [col for col in required_columns if col in df.columns] +
["Player_Team"]

    # Tạo cột Player_Team
    df["Player_Team"] = df["Player"].astype(str) + "_" + df["Team"].astype(str)

    # Chuyển đổi tuổi
    if "Age" in df.columns:
        df["Age"] = df["Age"].apply(parse_age)

    # Lưu DataFrame
    df = df[required_for_table]
    all_tables[table_id] = df
```

- **Ý nghĩa**:
  - Tải từng trang web bằng Selenium, chờ 2 giây để trang tải hoàn tất.
  - Dùng BeautifulSoup tìm bảng HTML theo table_id, chuyển thành DataFrame.
  - Đổi tên cột, kiểm tra cột Player và Team, loại bỏ cột trùng lặp.
  - Tạo cột Player_Team làm khóa gộp, chuẩn hóa tuổi, và lưu DataFrame vào all_tables.
- **Mục đích**: Thu thập dữ liệu từ các bảng trên trang web, chuẩn bị cho việc gộp.

7. Gộp, xử lý dữ liệu và lưu kết quả

```
# Xử lý dữ liệu
if all_tables:
    try:
        # Gộp các DataFrame
        merged_df = None
        for table_id, df in all_tables.items():
            if merged_df is None:
                merged_df = df.copy()
            else:
                df = df.drop_duplicates(subset=["Player_Team"], keep="first")
                merged_df = pd.merge(
                    merged_df,
                    df,
                    on=["Player_Team"],
                    how="outer",
                    suffixes=("", f"_{table_id}")
                )

                # Xử lý cột trùng lặp
                for col in required_columns:
```

```
                    if col == "Player_Team":
                        continue
                    col_duplicate = f"{col}_{table_id}"
                    if col in merged_df.columns and col_duplicate in
merged_df.columns:
                        merged_df[col] =
merged_df[col].combine_first(merged_df[col_duplicate])
                        merged_df = merged_df.drop(columns=[col_duplicate])

        if merged_df is None or merged_df.empty:
            raise ValueError("Không có dữ liệu hợp lệ để gộp.")

        # Xóa cột tạm thời
        if "Player_Team" in merged_df.columns:
            merged_df = merged_df.drop(columns=["Player_Team"])

        # Đảm bảo tất cả cột yêu cầu có mặt
        for col in required_columns:
            if col not in merged_df.columns:
                merged_df[col] = "N/A"

        # Sắp xếp lại cột
        merged_df = merged_df[required_columns]

        # Định nghĩa kiểu dữ liệu
        int_columns = [
            "Matches Played", "Starts", "Minutes", "Gls", "Ast", "crdY", "crdR",
            "PrgC", "PrgP", "PrgR", "Cmp", "KP", "Pass into 1_3", "PPA", "CrsPA",
            "ProDist", "TotDist", "Tkl", "TklW", "Deff Att", "Lost", "Blocks",
"Sh",
            "Pass", "Int", "Touches", "Def Pen", "Def 3rd", "Mid 3rd", "Att 3rd",
            "Att Pen", "Take-Ons Att", "Carries", "Carries 1_3", "CPA", "Mis",
"Dis",
            "Rec", "Rec PrgR", "Fls", "Fld", "Off", "Crs", "Recov", "Aerl Won",
"Aerl Lost"
        ]

        float_columns = [
            "Age", "xG", "xAG", "Gls per 90", "Ast per 90", "xG per 90", "xAG per
90",
            "GA90", "Save%", "CS%", "PK Save%", "SoT%", "SoT per 90", "G per Sh",
"Dist",
            "Cmp%", "ShortCmp%", "MedCmp%", "LongCmp%", "SCA", "SCA90", "GCA",
"GCA90",
```

```
            "Succ%", "Tkld%", "Aerl Won%"
        ]

        # Áp dụng kiểu dữ liệu
        for col in int_columns:
            if col in merged_df.columns and merged_df[col].dtype != "Int64":
                merged_df[col] = pd.to_numeric(merged_df[col],
errors="coerce").astype("Int64")

        for col in float_columns:
            if col in merged_df.columns and merged_df[col].dtype != float:
                merged_df[col] = pd.to_numeric(merged_df[col],
errors="coerce").round(2)

        # Lọc cầu thủ chơi trên 90 phút
        if "Minutes" in merged_df.columns:
            merged_df = merged_df[merged_df["Minutes"] > 90]

        # Làm sạch cột Nation
        if "Nation" in merged_df.columns:
            merged_df["Nation"] = merged_df["Nation"].apply(extract_country_code)

        # Sắp xếp theo tên cầu thủ
        if "Player" in merged_df.columns:
            merged_df = merged_df.sort_values(by="Player")

        # Lưu vào file CSV
        os.makedirs(output_dir, exist_ok=True)
        result_path = os.path.join(output_dir, "results.csv")

        merged_df.to_csv(result_path, index=False, encoding="utf-8-sig",
na_rep="N/A")
        print(f"✅ Lưu dữ liệu thành công vào {result_path} với
{merged_df.shape[0]} hàng và {merged_df.shape[1]} cột.")

    except Exception as e:
        print(f"Lỗi khi xử lý dữ liệu: {e}")
        print("❌ Không thể xử lý dữ liệu.")
else:
    print("❌ Không thu thập được dữ liệu.")

# Dọn dẹp
driver.quit()
```

- **Ý nghĩa**:
  - Gộp các DataFrame từ all_tables bằng pd.merge với khóa Player_Team, sử dụng outer để giữ toàn bộ dữ liệu.
  - Xử lý cột trùng lặp, đảm bảo tất cả cột trong required_columns đều có mặt.
  - Chuẩn hóa kiểu dữ liệu: số nguyên (Int64) cho các cột như Minutes, Gls; số thực (float) cho các cột như Age, Save%.
  - Lọc cầu thủ chơi trên 90 phút, chuẩn hóa quốc tịch, sắp xếp theo tên cầu thủ.
  - Lưu kết quả vào results.csv trong thư mục output, in thông báo thành công.
  - Đóng trình duyệt (driver.quit) để giải phóng tài nguyên.
- **Mục đích**: Tạo file results.csv chứa dữ liệu thống kê đầy đủ, sạch, và đúng định dạng.

Bài 2: Xử lý và gộp dữ liệu thống kê cầu thủ từ nhiều bảng, lưu vào results.csv.

1. Kiểm tra dữ liệu đã thu thập

```
# Xử lý dữ liệu
if all_tables:
    try:
        # Gộp các DataFrame
        merged_df = None
        for table_id, df in all_tables.items():
            if merged_df is None:
                merged_df = df.copy()
            else:
                df = df.drop_duplicates(subset=["Player_Team"], keep="first")
                merged_df = pd.merge(
                    merged_df,
                    df,
                    on=["Player_Team"],
                    how="outer",
                    suffixes=("", f"_{table_id}")
                )
```

- **Ý nghĩa**:
  - Kiểm tra xem từ điển all_tables (chứa các DataFrame từ các bảng đã crawl) có dữ liệu hay không.
  - Bắt đầu gộp các DataFrame:
    - DataFrame đầu tiên được sao chép làm merged_df.
    - Các DataFrame tiếp theo được gộp với merged_df bằng pd.merge, sử dụng cột Player_Team làm khóa.
    - Loại bỏ trùng lặp trong Player_Team trước khi gộp, dùng how="outer" để giữ tất cả dữ liệu.
- **Mục đích**: Tạo một DataFrame thống nhất từ nhiều bảng dữ liệu, đảm bảo không bỏ sót thông tin.

2. Xử lý cột trùng lặp sau khi gộp

```
# Xử lý cột trùng lặp
for col in required_columns:
    if col == "Player_Team":
        continue
    col_duplicate = f"{col}_{table_id}"
    if col in merged_df.columns and col_duplicate in merged_df.columns:
        merged_df[col] = merged_df[col].combine_first(merged_df[col_duplicate])
        merged_df = merged_df.drop(columns=[col_duplicate])
```

- **Ý nghĩa**:
  - Duyệt qua các cột trong required_columns để xử lý trường hợp cột trùng lặp (VD: Gls và Gls_stats_passing).
  - Nếu cột gốc (col) và cột trùng lặp (col_duplicate) cùng tồn tại, kết hợp dữ liệu bằng combine_first (ưu tiên giá trị không rỗng của cột gốc).
  - Xóa cột trùng lặp sau khi kết hợp.
- **Mục đích**: Loại bỏ các cột dư thừa sau khi gộp, đảm bảo mỗi cột chỉ xuất hiện một lần với dữ liệu chính xác.

3. Kiểm tra và xóa cột tạm thời

```
if merged_df is None or merged_df.empty:
    raise ValueError("Không có dữ liệu hợp lệ để gộp.")

# Xóa cột tạm thời
if "Player_Team" in merged_df.columns:
    merged_df = merged_df.drop(columns=["Player_Team"])
```

- **Ý nghĩa**:
  - Kiểm tra xem merged_df có dữ liệu hay không; nếu rỗng, báo lỗi và dừng xử lý.
  - Xóa cột Player_Team (được tạo để làm khóa gộp) vì không cần trong file kết quả.
- **Mục đích**: Đảm bảo dữ liệu hợp lệ và loại bỏ cột tạm thời để giữ DataFrame gọn gàng.

4. Đảm bảo đầy đủ cột yêu cầu

```
# Đảm bảo tất cả cột yêu cầu có mặt
for col in required_columns:
    if col not in merged_df.columns:
        merged_df[col] = "N/A"

# Sắp xếp lại cột
merged_df = merged_df[required_columns]
```

- **Ý nghĩa**:
  - Duyệt qua required_columns để kiểm tra các cột cần thiết.
  - Nếu cột nào thiếu, thêm cột đó với giá trị mặc định là "N/A".

    - Sắp xếp lại các cột trong merged_df theo thứ tự của required_columns.
- **Mục đích**: Đảm bảo file kết quả có đầy đủ các cột mong muốn, đúng thứ tự, ngay cả khi dữ liệu gốc thiếu.

5. Chuẩn hóa kiểu dữ liệu

```
# Định nghĩa kiểu dữ liệu
int_columns = [
    "Matches Played", "Starts", "Minutes", "Gls", "Ast", "crdY", "crdR",
    "PrgC", "PrgP", "PrgR", "Cmp", "KP", "Pass into 1_3", "PPA", "CrsPA",
    "ProDist", "TotDist", "Tkl", "TklW", "Deff Att", "Lost", "Blocks", "Sh",
    "Pass", "Int", "Touches", "Def Pen", "Def 3rd", "Mid 3rd", "Att 3rd",
    "Att Pen", "Take-Ons Att", "Carries", "Carries 1_3", "CPA", "Mis", "Dis",
    "Rec", "Rec PrgR", "Fls", "Fld", "Off", "Crs", "Recov", "Aerl Won", "Aerl
Lost"
]

float_columns = [
    "Age", "xG", "xAG", "Gls per 90", "Ast per 90", "xG per 90", "xAG per 90",
    "GA90", "Save%", "CS%", "PK Save%", "SoT%", "SoT per 90", "G per Sh", "Dist",
    "Cmp%", "ShortCmp%", "MedCmp%", "LongCmp%", "SCA", "SCA90", "GCA", "GCA90",
    "Succ%", "Tkld%", "Aerl Won%"
]

# Áp dụng kiểu dữ liệu
for col in int_columns:
    if col in merged_df.columns and merged_df[col].dtype != "Int64":
        merged_df[col] = pd.to_numeric(merged_df[col],
errors="coerce").astype("Int64")

for col in float_columns:
    if col in merged_df.columns and merged_df[col].dtype != float:
        merged_df[col] = pd.to_numeric(merged_df[col], errors="coerce").round(2)
```

- **Ý nghĩa**:
    - Xác định danh sách cột kiểu số nguyên (int_columns) và số thực (float_columns).
    - Chuyển đổi kiểu dữ liệu:
        - Cột số nguyên (VD: Minutes, Gls) thành Int64 (hỗ trợ giá trị NaN).
        - Cột số thực (VD: Age, Save%) thành float, làm tròn 2 chữ số thập phân.
    - Dữ liệu không hợp lệ được chuyển thành NaN.
- **Mục đích**: Chuẩn hóa kiểu dữ liệu để đảm bảo tính nhất quán và phù hợp cho phân tích sau này.

6. Lọc và làm sạch dữ liệu

```
# Lọc cầu thủ chơi trên 90 phút
```

```
if "Minutes" in merged_df.columns:
    merged_df = merged_df[merged_df["Minutes"] > 90]

# Làm sạch cột Nation
if "Nation" in merged_df.columns:
    merged_df["Nation"] = merged_df["Nation"].apply(extract_country_code)

# Sắp xếp theo tên cầu thủ
if "Player" in merged_df.columns:
    merged_df = merged_df.sort_values(by="Player")
```

- **Ý nghĩa**:
  - Lọc các cầu thủ có thời gian thi đấu trên 90 phút để tập trung vào những người có đóng góp đáng kể.
  - Chuẩn hóa cột Nation bằng hàm extract_country_code (VD: "England ENG" -> "ENG").
  - Sắp xếp DataFrame theo tên cầu thủ (Player) để dễ tra cứu.
- **Mục đích**: Tinh chỉnh dữ liệu, loại bỏ cầu thủ ít thi đấu, chuẩn hóa quốc tịch, và sắp xếp kết quả.

7. Lưu kết quả và dọn dẹp

```
# Lưu vào file CSV
os.makedirs(output_dir, exist_ok=True)
result_path = os.path.join(output_dir, "results.csv")

merged_df.to_csv(result_path, index=False, encoding="utf-8-sig", na_rep="N/A")
print(f"✅ Lưu dữ liệu thành công vào {result_path} với {merged_df.shape[0]} hàng và {merged_df.shape[1]} cột.")

except Exception as e:
    print(f"Lỗi khi xử lý dữ liệu: {e}")
    print("❌ Không thể xử lý dữ liệu.")
else:
    print("❌ Không thu thập được dữ liệu.")

# Dọn dẹp
driver.quit()
```

- **Ý nghĩa**:
  - Tạo thư mục output nếu chưa tồn tại (exist_ok=True).
  - Lưu merged_df vào results.csv với mã hóa utf-8-sig (hỗ trợ tiếng Việt) và thay giá trị thiếu bằng "N/A".
  - In thông báo thành công với số hàng và cột của file.
  - Xử lý lỗi (nếu có) và thông báo nếu không thu thập được dữ liệu.
  - Đóng trình duyệt (driver.quit) để giải phóng tài nguyên.
- **Mục đích**: Xuất dữ liệu cuối cùng vào file CSV, thông báo kết quả, và dọn dẹp tài nguyên.

BÀI 3: Lọc cầu thủ thi đấu trên 900 phút từ results.csv, crawl giá trị chuyển nhượng từ footballtransfers.com, và lưu vào player_transfer_fee.csv.

1. Import thư viện

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import seaborn as sns
import os
import sys
```

- **Ý nghĩa**:
  - Nhập các thư viện cần thiết:
    - pandas, numpy: Xử lý dữ liệu bảng và số học.
    - matplotlib.pyplot, seaborn: Vẽ biểu đồ (Elbow, Silhouette, PCA).
    - sklearn.decomposition.PCA: Giảm chiều dữ liệu để trực quan hóa.
    - sklearn.cluster.KMeans: Phân cụm dữ liệu.
    - sklearn.metrics.silhouette_score: Đánh giá chất lượng cụm.
    - os, sys: Quản lý file và tham số dòng lệnh.
- **Mục đích**: Chuẩn bị công cụ cho việc đọc dữ liệu, phân cụm, đánh giá, và trực quan hóa.

2. Thiết lập đường dẫn file và danh sách cột số

```
# Đường dẫn file mặc định
DEFAULT_CSV_FILE = r"c:\Users\hp\csv\results.csv"


# Danh sách cột số (49 cột hợp lệ)
EXPECTED_NUMERIC_COLUMNS = [
    'Age', 'Matches Played', 'Starts', 'Minutes', 'Gls', 'Ast', 'crdY', 'crdR',
    'xG', 'xAG', 'PrgC', 'PrgP', 'PrgR', 'Gls per 90', 'Ast per 90', 'xG per 90',
    'xAG per 90', 'KP', 'PPA', 'CrsPA', 'SCA', 'SCA90', 'GCA', 'GCA90', 'Tkl',
    'TklW', 'Blocks', 'Sh', 'Pass', 'Int', 'Touches', 'Succ%', 'Carries',
'TotDist',
    'CPA', 'Mis', 'Dis', 'Rec', 'Fls', 'Fld', 'Off', 'Crs', 'Cmp', 'Cmp%',
    'Aerl Won', 'Aerl Lost', 'Aerl Won%'
]
```

- **Ý nghĩa**:
  - Định nghĩa đường dẫn mặc định đến file results.csv chứa dữ liệu thống kê cầu thủ.
  - Liệt kê 49 cột số (EXPECTED_NUMERIC_COLUMNS) đại diện cho các chỉ số thống kê (tuổi, bàn thắng, chuyền bóng, phòng ngự, v.v.) dùng cho phân cụm.
- **Mục đích**: Xác định nguồn dữ liệu và các đặc trưng số để phân cụm.

3. Hàm đọc dữ liệu

```
# Hàm đọc dữ liệu
def read_data(file_path):
    if not os.path.exists(file_path):
        print(f"Error: File '{file_path}' does not exist.")
        sys.exit(1)

    encodings = ['utf-8-sig', 'latin1']
    df_raw = None
    for enc in encodings:
        try:
            df_raw = pd.read_csv(file_path, encoding=enc)
            print(f"Reading file '{file_path}' successfully with encoding
'{enc}'.")
            break
        except Exception as e:
            print(f"Failed with encoding '{enc}': {e}")

    if df_raw is None:
        print("Error: Could not read file with any encoding.")
        sys.exit(1)

    return df_raw.copy()
```

- **Ý nghĩa**:
  - Kiểm tra sự tồn tại của file CSV tại file_path.
  - Thử đọc file với các mã hóa (utf-8-sig, latin1) để xử lý vấn đề định dạng.
  - Báo lỗi và thoát nếu không đọc được file, trả về bản sao DataFrame nếu thành công.
- **Mục đích**: Đọc dữ liệu từ results.csv một cách an toàn, hỗ trợ nhiều mã hóa.

4. Hàm lấy và chuẩn hóa cột số

```
# Hàm lấy cột số
def get_numeric_columns(df, expected_columns):
    numeric_columns = []
    for col in expected_columns:
        if col in df.columns:
            try:
                df[col] = pd.to_numeric(df[col], errors='coerce')
                if not df[col].isna().all():
                    numeric_columns.append(col)
                else:
                    print(f"Column '{col}' contains only NaN after conversion.
Skipping.")
            except Exception as e:
```

```
                print(f"Column '{col}' cannot be converted to numeric: {e}.
Skipping.")
        else:
            print(f"Column '{col}' not found in DataFrame. Skipping.")

    if not numeric_columns:
        print("Error: No valid numeric columns found.")
        sys.exit(1)

    return numeric_columns

# Hàm chuẩn hóa thủ công
def manual_standardize(data):
    scaled_data = data.copy()
    for col in data.columns:
        mean = data[col].mean(skipna=True)
        std = data[col].std(skipna=True)
        if pd.isna(std) or std == 0:
            print(f"Column '{col}' has zero or NaN standard deviation. Setting
standardized values to 0.")
            scaled_data[col] = 0
        else:
            scaled_data[col] = (data[col] - mean) / std
        scaled_data[col] = scaled_data[col].fillna(0)
    return scaled_data.values
```

- **Ý nghĩa**:
  - get_numeric_columns: Lọc các cột số hợp lệ từ expected_columns, chuyển đổi sang kiểu số, bỏ qua cột chỉ chứa NaN hoặc không thể chuyển đổi.
  - manual_standardize: Chuẩn hóa dữ liệu thủ công (trừ trung bình, chia độ lệch chuẩn), thay NaN bằng 0, xử lý trường hợp độ lệch chuẩn bằng 0.
- **Mục đích**: Chuẩn bị dữ liệu số sạch và chuẩn hóa để phân cụm, đảm bảo không có lỗi do giá trị thiếu hoặc không hợp lệ.

5. Hàm vẽ biểu đồ PCA và phân cụm

```
# Hàm vẽ biểu đồ PCA (bỏ tâm cụm)
def plot_pca_clusters(scaled_data, df, k, output_file):
    kmeans = KMeans(n_clusters=k, random_state=42)
    df[f'Cluster_k{k}'] = kmeans.fit_predict(scaled_data)

    pca = PCA(n_components=2)
    pca_components = pca.fit_transform(scaled_data)
    explained_variance = pca.explained_variance_ratio_.sum()
```

```
    df_pca = pd.DataFrame(pca_components, columns=['PCA1', 'PCA2'])
    df_pca['Cluster'] = df[f'Cluster_k{k}']

    plt.figure(figsize=(12, 8))
    sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=df_pca,
palette='viridis', s=100, alpha=0.7)
    plt.title(f"Player Clustering with k={k} (PCA 2D, Explained Variance:
{explained_variance:.2%})")
    plt.xlabel("PCA 1")
    plt.ylabel("PCA 2")
    plt.legend(title="Cluster")
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(output_file)
    plt.show()

    print(f"\nMean Statistics for k={k}:")
    cluster_summary =
df.groupby(f'Cluster_k{k}')[EXPECTED_NUMERIC_COLUMNS].mean().round(2)
    print(cluster_summary)

    print(f"\nTop 5 Players per Cluster for k={k}:")
    for cluster in range(k):
        cluster_players = df[df[f'Cluster_k{k}'] == cluster][['Player',
'Team']].head(5)
        print(f"\nCluster {cluster}:")
        print(cluster_players.to_string(index=False))
```

- **Ý nghĩa**:
  - Phân cụm dữ liệu bằng K-means với k cụm, lưu nhãn cụm vào cột Cluster_k{k}.
  - Giảm chiều dữ liệu xuống 2D bằng PCA để trực quan hóa, tính tỷ lệ phương sai giải thích (explained_variance).
  - Vẽ biểu đồ phân tán với seaborn, hiển thị các cụm, lưu vào file output_file.
  - In thống kê trung bình của các cột số theo cụm và danh sách 5 cầu thủ đầu tiên trong mỗi cụm.
- **Mục đích**: Trực quan hóa và phân tích kết quả phân cụm, giúp hiểu đặc điểm của từng cụm.

6. Hàm chính: Phân cụm và đánh giá số cụm

```
# Hàm chính
def main():
    csv_file = sys.argv[1] if len(sys.argv) > 1 else DEFAULT_CSV_FILE

    df = read_data(csv_file)
```

```
    numeric_columns = get_numeric_columns(df, EXPECTED_NUMERIC_COLUMNS)
    print(f"Selected {len(numeric_columns)} numeric columns: {numeric_columns}")

    data_for_clustering = df[numeric_columns]
    scaled_data = manual_standardize(data_for_clustering)

    # Elbow Method và Silhouette Score
    inertia = []
    silhouette_scores = []
    k_range = range(2, 11)

    for k in k_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_data)
        inertia.append(kmeans.inertia_)
        if k > 1:
            score = silhouette_score(scaled_data, kmeans.labels_)
            silhouette_scores.append(score)
        else:
            silhouette_scores.append(0)

    # Vẽ Elbow Method
    plt.figure(figsize=(10, 6))
    plt.plot(k_range, inertia, marker='o', color='blue', label='Inertia')
    plt.axvline(x=5, color='red', linestyle='--', label='k=5')
    plt.title("Elbow Method for Optimal Number of Clusters")
    plt.xlabel("Number of Clusters (k)")
    plt.ylabel("Inertia")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig("elbow_plot.png")
    plt.close()

    # Vẽ Silhouette Score
    plt.figure(figsize=(10, 6))
    plt.plot(k_range, silhouette_scores, marker='o', color='green',
label='Silhouette Score')
    plt.axvline(x=5, color='red', linestyle='--', label='k=5')
    plt.title("Silhouette Score for Different Numbers of Clusters")
    plt.xlabel("Number of Clusters (k)")
    plt.ylabel("Silhouette Score")
    plt.legend()
    plt.grid(True)
```

```
plt.tight_layout()
plt.savefig("silhouette_plot.png")
plt.close()

# Phân cụm và trực quan cho k=5
print("\n=== Clustering with k=5 ===")
plot_pca_clusters(scaled_data, df, k=5, output_file="cluster_plot_k5.png")
```

- **Ý nghĩa**:
  - Đọc file CSV, chọn cột số, chuẩn hóa dữ liệu.
  - Dùng Elbow Method (đánh giá inertia) và Silhouette Score để tìm số cụm tối ưu (k từ 2 đến 10).
  - Vẽ biểu đồ Elbow (elbow_plot.png) và Silhouette (silhouette_plot.png), đánh dấu k=5 làm lựa chọn mặc định.
  - Thực hiện phân cụm với k=5 và gọi hàm plot_pca_clusters để trực quan hóa.
- **Mục đích**: Phân cụm cầu thủ, chọn số cụm tối ưu, và trực quan hóa kết quả.

7. Bình luận kết quả phân cụm

```
# Bình luận
print("\nComments on Clustering Results:")
print("1. **Choice of k=5 vs k=3**:")
print("   - The Elbow Method likely shows a bend at k=3 or k=4, but k=5 provides more granularity, capturing distinct player roles.")
print("   - Silhouette Score for k=5 (~0.2-0.3) may be lower than k=3 (~0.3-0.4), indicating slightly less distinct clusters, but still acceptable for detailed analysis.")
print("   - k=5 is suitable for separating roles like center-backs, goalkeepers, midfielders, forwards, and low-contribution players.")
print("   - k=3 is simpler, grouping into defensive, attacking, and low-contribution, but may miss nuanced role differences.")
print("2. **Cluster Interpretation (k=5)**:")
print("   - Cluster 0: Center-backs (high Tkl, Blocks, Minutes, e.g., Virgil van Dijk).")
print("   - Cluster 1: Goalkeepers (low Gls, high Cmp%, e.g., Łukasz Fabiański).")
print("   - Cluster 2: Midfielders (high Pass, Ast, SCA, e.g., Bruno Fernandes).")
print("   - Cluster 3: Forwards (high Gls, xG, e.g., Mohamed Salah).")
print("   - Cluster 4: Low-contribution players (low Minutes, Gls, e.g., Ayden Heaven).")
print("3. **Visualization**:")
print("   - Explained variance (~30-50%) is sufficient for visualization but indicates some information loss.")
print("   - Some overlap may occur due to dimensionality reduction, but clusters remain meaningful based on K-means in the original feature space.")
```

```
print("4. **Recommendation**:")
print("   - k=5 is recommended for detailed role segmentation (e.g., separating goalkeepers from defenders).")
print("   - Revert to k=3 for simpler, more interpretable clusters if k=5 shows excessive overlap or low Silhouette Score.")
print("   - Check Silhouette Score in 'silhouette_plot.png': if k=5 score is <0.2, consider k=3 or k=4 for better separation.")


if __name__ == "__main__":
    main()
```

- **Ý nghĩa**:
  - Cung cấp bình luận chi tiết về kết quả phân cụm:
    - So sánh k=5 và k=3 dựa trên Elbow Method và Silhouette Score.
    - Giải thích ý nghĩa của từng cụm với k=5 (hậu vệ, thủ môn, tiền vệ, tiền đạo, cầu thủ ít đóng góp).
    - Nhận xét về trực quan hóa PCA và khuyến nghị lựa chọn k.
  - Chạy hàm main khi script được gọi trực tiếp.
- **Mục đích**: Giải thích kết quả phân cụm, giúp người dùng hiểu ý nghĩa các cụm và đưa ra khuyến nghị.

BÀI 4 YÊU CẦU 1: Ước lượng giá trị chuyển nhượng của cầu thủ dựa trên thống kê, lưu kết quả vào ml_transfer_values_gradient.csv.

1. Import thư viện

```
import pandas as pd  # Xử lý dữ liệu CSV
import os  # Làm việc với đường dẫn file
from fuzzywuzzy import fuzz, process  # So khớp tên cầu thủ
from selenium import webdriver  # Tự động hóa trình duyệt
from selenium.webdriver.chrome.service import Service  # Quản lý ChromeDriver
from selenium.webdriver.chrome.options import Options  # Cấu hình Chrome
from selenium.webdriver.common.by import By  # Tìm kiếm phần tử HTML
from selenium.webdriver.support.ui import WebDriverWait  # Chờ tải trang
from selenium.webdriver.support import expected_conditions as EC  # Điều kiện chờ
from webdriver_manager.chrome import ChromeDriverManager  # Tự động tải ChromeDriver
import time  # Dùng để tạm dừng khi thử lại
from selenium.common.exceptions import WebDriverException  # Xử lý lỗi Selenium
```

- **Ý nghĩa**:
  - Nhập các thư viện cần thiết:
    - pandas: Xử lý dữ liệu CSV.
    - os: Quản lý đường dẫn file.
    - fuzzywuzzy: So khớp tên cầu thủ.
    - selenium và các module liên quan: Tự động hóa trình duyệt, chờ tải trang, xử lý lỗi.
    - time: Tạm dừng khi thử lại crawl.

    - webdriver_manager: Tự động tải ChromeDriver.
- **Mục đích**: Chuẩn bị công cụ để đọc dữ liệu, crawl web, so khớp tên, và lưu kết quả.

2. Hàm lấy đường dẫn thư mục CSV

```
# Hàm lấy đường dẫn thư mục csv
def get_csv_dir():
    """Tạo đường dẫn đến thư mục csv, nằm ngoài thư mục chứa script."""
    base_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), "..",
"..."))
    return os.path.join(base_dir, "csv")
```

- **Ý nghĩa**:
    - Tạo đường dẫn đến thư mục csv nằm ngoài thư mục chứa script (cụ thể là c:\Users\hp\csv).
    - Sử dụng os.path để đảm bảo đường dẫn tương thích trên các hệ điều hành.
- **Mục đích**: Cung cấp đường dẫn chuẩn để đọc và lưu file CSV (results.csv, players_over_900_minutes.csv, player_transfer_fee.csv).

3. Hàm lọc cầu thủ từ results.csv

```
# Hàm lọc cầu thủ từ results.csv
def filter_players(input_path, output_path):
    """
    Đọc file results.csv, lọc cầu thủ có >900 phút, lưu vào file mới.
    Input: Đường dẫn file results.csv
    Output: Đường dẫn file players_over_900_minutes.csv
    """
    # Đọc file CSV, coi "N/A" là giá trị NaN
    df = pd.read_csv(input_path, na_values=["N/A"])
    # Lọc cầu thủ có thời gian thi đấu >900 phút
    filtered_df = df[df['Minutes'] > 900].copy()
    print(f"Số cầu thủ có trên 900 phút: {len(filtered_df)}")
    # Lưu file CSV mới
    filtered_df.to_csv(output_path, index=False, encoding='utf-8-sig')
    print(f"Đã lưu danh sách cầu thủ vào: {output_path}")
    return filtered_df
```

- **Ý nghĩa**:
    - Đọc file results.csv, chuyển "N/A" thành NaN.
    - Lọc các cầu thủ có thời gian thi đấu trên 900 phút.
    - Lưu danh sách cầu thủ đã lọc vào file players_over_900_minutes.csv với mã hóa utf-8-sig.
    - In số lượng cầu thủ và thông báo lưu file.
- **Mục đích**: Tạo danh sách cầu thủ chính (thi đấu đáng kể) để crawl giá trị chuyển nhượng.

4. Hàm rút gọn tên cầu thủ

```
# Hàm rút gọn tên cầu thủ
def shorten_name(name):
    """Rút gọn tên cầu thủ thành 2 từ đầu tiên để so khớp."""
    parts = name.strip().split()
    return " ".join(parts[:2]) if len(parts) >= 2 else name
```

- **Ý nghĩa**:
  - Rút gọn tên cầu thủ bằng cách lấy 2 từ đầu tiên (VD: "Mohamed Salah Ahmed" -> "Mohamed Salah").
  - Nếu tên chỉ có một từ, giữ nguyên.
- **Mục đích**: Chuẩn hóa tên cầu thủ để dễ so khớp với dữ liệu crawl từ web, tránh sai lệch do tên dài hoặc định dạng khác nhau.

5. Hàm crawl dữ liệu chuyển nhượng

```
# Hàm crawl dữ liệu chuyển nhượng
def scrape_transfer_data(driver, urls, player_names):
    """
    Crawl dữ liệu chuyển nhượng từ danh sách URL, khớp với danh sách cầu thủ.
    Input: driver (ChromeDriver), danh sách URL, danh sách tên cầu thủ
    Output: Danh sách [tên cầu thủ, giá chuyển nhượng]
    """
    data = []
    seen_players = set()  # Lưu tên cầu thủ đã xử lý để tránh trùng lặp

    for url in urls:
        for attempt in range(3):  # Thử tối đa 3 lần nếu lỗi
            try:
                print(f"Đang crawl: {url}")
                driver.get(url)
                # Chờ bảng chuyển nhượng xuất hiện (tối đa 10 giây)
                table = WebDriverWait(driver, 10).until(
                    EC.presence_of_element_located((By.CLASS_NAME, "transfer-
table"))
                )
                # Lấy tất cả các hàng trong bảng
                rows = table.find_elements(By.TAG_NAME, "tr")

                # Duyệt từng hàng
                for row in rows:
                    cols = row.find_elements(By.TAG_NAME, "td")
                    if cols and len(cols) >= 3:  # Đảm bảo có đủ cột
                        # Lấy tên cầu thủ
                        player_name = cols[0].text.strip().split("\n")[0].strip()
```

```
                        # Rút gọn tên để so khớp
                        shortened_name = shorten_name(player_name)
                        # Lấy giá trị chuyển nhượng
                        transfer_value = cols[-1].text.strip() if cols[-
1].text.strip() else "N/A"

                        # Chỉ xử lý nếu có giá trị chuyển nhượng hợp lệ
                        if transfer_value != "N/A":
                            # So khớp tên với danh sách cầu thủ
                            best_match = process.extractOne(
                                shortened_name, player_names,
scorer=fuzz.token_sort_ratio
                            )
                            if best_match and best_match[1] >= 90:  # Ngưỡng so
khớp 90%
                                if player_name not in seen_players:
                                    seen_players.add(player_name)
                                    data.append([player_name, transfer_value])
                                    print(f"Khớp: {player_name} (Giá:
{transfer_value})")
                break  # Thoát vòng lặp retry nếu crawl thành công
            except WebDriverException as e:
                print(f"Lỗi khi crawl {url} (lần {attempt + 1}/3): {e}")
                time.sleep(2)  # Chờ 2 giây trước khi thử lại
                if attempt == 2:
                    print(f"Không thể crawl {url} sau 3 lần thử.")

    return data
```

- **Ý nghĩa**:
  - Crawl dữ liệu từ danh sách URL (bảng chuyển nhượng trên footballtransfers.com).
  - Sử dụng Selenium để tải trang, chờ bảng transfer-table xuất hiện (10 giây).
  - Lấy tên cầu thủ và giá trị chuyển nhượng từ mỗi hàng bảng.
  - Rút gọn tên, so khớp với danh sách cầu thủ bằng fuzzywuzzy (ngưỡng 90%).
  - Tránh trùng lặp bằng seen_players, thử lại tối đa 3 lần nếu lỗi.
- **Mục đích**: Thu thập dữ liệu giá trị chuyển nhượng của các cầu thủ khớp với danh sách đã lọc.

6. Hàm chính (main)

```
# Hàm chính để chạy chương trình
def main():
    """Chạy toàn bộ quy trình: lọc cầu thủ, crawl dữ liệu, lưu kết quả."""
    # Lấy đường dẫn thư mục csv
    csv_dir = get_csv_dir()
    result_path = os.path.join(csv_dir, "results.csv")
```

```
    filtered_path = os.path.join(csv_dir, "players_over_900_minutes.csv")
    output_path = os.path.join(csv_dir, "player_transfer_fee.csv")

    # Lọc cầu thủ từ results.csv
    filtered_df = filter_players(result_path, filtered_path)

    # Tạo danh sách tên cầu thủ rút gọn
    player_names = [shorten_name(name) for name in
filtered_df['Player'].str.strip()]

    # Cấu hình ChromeDriver
    options = Options()
    options.add_argument("--no-sandbox")  # Tắt sandbox để tránh lỗi
    options.add_argument("--disable-dev-shm-usage")  # Tắt bộ nhớ chia sẻ
    options.add_argument("--ignore-certificate-errors")  # Bỏ qua lỗi SSL
    options.add_argument("--allow-insecure-localhost")  # Cho phép kết nối không
an toàn
    # Tạm thời tắt headless để kiểm tra lỗi SSL
    # options.add_argument("--headless=new")  # Có thể bật lại sau khi sửa lỗi

    # Khởi tạo ChromeDriver
    driver = webdriver.Chrome(
        service=Service(ChromeDriverManager().install()), options=options
    )

    # Tạo danh sách URL (crawl 10 trang để đảm bảo đủ dữ liệu)
    base_url = "https://www.footballtransfers.com/us/transfers/confirmed/2024-
2025/uk-premier-league/"
    urls = [f"{base_url}{i}" for i in range(1, 11)]

    # Crawl dữ liệu chuyển nhượng
    data = scrape_transfer_data(driver, urls, player_names)

    # Đóng trình duyệt
    driver.quit()
```

- **Ý nghĩa**:
  - Thiết lập đường dẫn cho các file: results.csv (đầu vào), players_over_900_minutes.csv (danh sách lọc), player_transfer_fee.csv (kết quả).
  - Lọc cầu thủ, tạo danh sách tên rút gọn.
  - Cấu hình ChromeDriver với các tùy chọn để xử lý lỗi SSL và tối ưu hiệu suất.
  - Tạo danh sách 10 URL để crawl dữ liệu chuyển nhượng.
  - Gọi hàm scrape_transfer_data để thu thập dữ liệu, đóng trình duyệt sau khi xong.
- **Mục đích**: Điều phối toàn bộ quy trình từ lọc dữ liệu, crawl, đến chuẩn bị lưu kết quả.

7. Lưu kết quả và chạy chương trình

```
# Lưu kết quả vào file CSV
if data:
    df_output = pd.DataFrame(data, columns=['Player', 'Price'])
    df_output.to_csv(output_path, index=False, encoding='utf-8-sig')
    print(f"Kết quả đã được lưu vào: {output_path} với {len(df_output)} cầu
thủ.")
else:
    print("Không tìm thấy cầu thủ nào khớp.")

# Chạy chương trình
if __name__ == "__main__":
    main()
```

- **Ý nghĩa**:
    - Nếu có dữ liệu từ scrape_transfer_data, tạo DataFrame với cột Player và Price, lưu vào player_transfer_fee.csv với mã hóa utf-8-sig.
    - In thông báo số lượng cầu thủ hoặc thông báo không tìm thấy dữ liệu.
    - Chạy hàm main khi script được gọi trực tiếp.
- **Mục đích**: Xuất kết quả crawl thành file CSV, cung cấp thông tin về dữ liệu thu thập được.

CÂU 4 Ý 2: Ước lượng ETV cầu thủ dựa trên thống kê, lưu vào ml_transfer_values_gradient.csv.

1. Import thư viện

```
import pandas as pd  # Xử lý dữ liệu CSV
import numpy as np  # Xử lý số học
import os  # Quản lý đường dẫn file
from fuzzywuzzy import fuzz, process  # So khớp tên cầu thủ
import re  # Xử lý chuỗi
from sklearn.model_selection import train_test_split  # Chia dữ liệu train/test
from sklearn.ensemble import GradientBoostingRegressor  # Mô hình Gradient
Boosting
from sklearn.preprocessing import StandardScaler  # Chuẩn hóa dữ liệu số
from sklearn.metrics import mean_squared_error, r2_score  # Đánh giá mô hình
from category_encoders import TargetEncoder  # Mã hóa đặc trưng phân loại
from sklearn.pipeline import Pipeline  # Xây dựng pipeline xử lý
```

- **Ý nghĩa**:
    - Nhập các thư viện cần thiết:
        - pandas, numpy: Xử lý dữ liệu và số học.
        - os, re: Quản lý file và xử lý chuỗi giá trị chuyển nhượng.
        - fuzzywuzzy: So khớp tên cầu thủ.

    - sklearn: Các công cụ học máy (chia dữ liệu, mô hình, chuẩn hóa, đánh giá).
    - category_encoders: Mã hóa đặc trưng phân loại (Team, Nation).
- **Mục đích**: Chuẩn bị công cụ để đọc dữ liệu, xử lý, huấn luyện mô hình, và lưu kết quả.

2. Cấu hình vị trí cầu thủ và đặc trưng

```
# Cấu hình các vị trí cầu thủ và đặc trưng
positions_config = {
    'GK': {
        'position_filter': 'GK',
        'features': ['Save%', 'CS%', 'GA90', 'PK Save%', 'Minutes', 'Age',
'Team', 'Nation']
    },
    'DF': {
        'position_filter': 'DF',
        'features': ['Tkl', 'Int', 'Blocks', 'Aerl Won%', 'Recov', 'Cmp%',
'PrgP', 'Minutes', 'Age', 'Team', 'Nation']
    },
    'MF': {
        'position_filter': 'MF',
        'features': ['Cmp%', 'KP', 'PPA', 'PrgP', 'SCA', 'xAG', 'Tkl', 'Ast',
'Minutes', 'Age', 'Team', 'Nation']
    },
    'FW': {
        'position_filter': 'FW',
        'features': ['Gls', 'Ast', 'Gls per 90', 'xG per 90', 'SCA90', 'GCA90',
'PrgC', 'Minutes', 'Age', 'Team', 'Nation']
    }
}
```

- **Ý nghĩa**:
    - Định nghĩa từ điển positions_config cho 4 vị trí (GK, DF, MF, FW).
    - Mỗi vị trí có:
        - position_filter: Bộ lọc vị trí (VD: 'GK').
        - features: Danh sách đặc trưng (thống kê, tuổi, đội, quốc tịch) phù hợp với vị trí.
- **Mục đích**: Xác định đặc trưng riêng cho từng vị trí để huấn luyện mô hình dự đoán ETV.

3. Hàm lấy đường dẫn và chuyển đổi giá trị

```
# Hàm lấy đường dẫn thư mục csv
def get_csv_dir():
    """Tạo đường dẫn đến thư mục csv."""
    base_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), "..",
".."))
    return os.path.join(base_dir, "csv")
```

```
# Hàm chuyển đổi giá trị chuyển nhượng (VD: "€2.5M" -> 2500000)
def parse_etv(etv_text):
    """Chuyển đổi chuỗi giá trị chuyển nhượng thành số."""
    if pd.isna(etv_text) or etv_text in ["N/A", ""]:
        return np.nan
    try:
        etv_text = re.sub(r'[€£]', '', etv_text).strip().upper()
        multiplier = 1000000 if 'M' in etv_text else 1000 if 'K' in etv_text else
1
        value = float(re.sub(r'[MK]', '', etv_text)) * multiplier
        return value
    except (ValueError, TypeError):
        return np.nan
```

- **Ý nghĩa**:
  - get_csv_dir: Tạo đường dẫn đến thư mục csv (VD: c:\Users\hp\csv).
  - parse_etv: Chuyển đổi giá trị chuyển nhượng từ chuỗi (VD: "€2.5M") thành số (2500000), trả về NaN nếu không hợp lệ.
- **Mục đích**: Cung cấp đường dẫn chuẩn và chuẩn hóa giá trị chuyển nhượng để xử lý dữ liệu.

4. Hàm so khớp và rút gọn tên cầu thủ

```
# Hàm so khớp tên cầu thủ
def fuzzy_match_name(name, choices, score_threshold=90):
    """Tìm tên gần giống nhất trong choices với ngưỡng tương đồng 90."""
    if not isinstance(name, str):
        return None, None
    shortened_name = shorten_name(name).lower()
    shortened_choices = [shorten_name(c).lower() for c in choices if
isinstance(c, str)]
    match = process.extractOne(
        shortened_name,
        shortened_choices,
        scorer=fuzz.token_sort_ratio,
        score_cutoff=score_threshold
    )
    if match:
        matched_idx = shortened_choices.index(match[0])
        return choices[matched_idx], match[1]
    return None, None

# Hàm rút gọn tên cầu thủ
def shorten_name(name):
    """Rút gọn tên thành 2 từ đầu tiên."""
```

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

- **Ý nghĩa**:
  - shorten_name: Rút gọn tên cầu thủ thành 2 từ đầu tiên (VD: "Mohamed Salah Ahmed" -> "Mohamed Salah").
  - fuzzy_match_name: So khớp tên cầu thủ với danh sách tên bằng fuzzywuzzy, yêu cầu độ tương đồng ≥90%.
- **Mục đích**: Chuẩn hóa và so khớp tên cầu thủ giữa results.csv và player_transfer_fee.csv, tránh sai lệch do định dạng tên.

5. Hàm xử lý dữ liệu và huấn luyện mô hình

```
# Hàm xử lý dữ liệu và huấn luyện mô hình cho mỗi vị trí
def process_position(position, config, results_path, etv_path):
    """
    Xử lý dữ liệu, huấn luyện mô hình, dự đoán ETV cho một vị trí.
    Input: vị trí (GK, DF, MF, FW), config, đường dẫn file results.csv và player_transfer_fee.csv
    Output: DataFrame kết quả, danh sách cầu thủ không khớp
    """
    # Đọc dữ liệu
    try:
        df_results = pd.read_csv(results_path)
        df_etv = pd.read_csv(etv_path)
    except FileNotFoundError as e:
        print(f"Lỗi: Không tìm thấy file cho {position} - {e}")
        return None, None

    # Lấy vị trí chính
    df_results['Primary_Position'] = df_results['Position'].astype(str).str.split(r'[,/]').str[0].str.strip()
    df_results = df_results[df_results['Primary_Position'].str.upper() == config['position_filter'].upper()].copy()

    # So khớp tên cầu thủ
    player_names = df_etv['Player'].dropna().tolist()
    df_results['Matched_Name'] = None
    df_results['Match_Score'] = None
    df_results['ETV'] = np.nan

    for idx, row in df_results.iterrows():
        matched_name, score = fuzzy_match_name(row['Player'], player_names)
```

```
        if matched_name:
            df_results.at[idx, 'Matched_Name'] = matched_name
            df_results.at[idx, 'Match_Score'] = score
            matched_row = df_etv[df_etv['Player'] == matched_name]
            if not matched_row.empty:
                df_results.at[idx, 'ETV'] =
parse_etv(matched_row['Price'].iloc[0])

    # Lọc dữ liệu đã khớp
    df_filtered =
df_results[df_results['Matched_Name'].notna()].drop_duplicates(subset='Matched_Na
me')
    unmatched =
df_results[df_results['Matched_Name'].isna()]['Player'].dropna().tolist()
    if unmatched:
        print(f"Cầu thủ {position} không khớp: {len(unmatched)} cầu thủ.")
        print(unmatched)

    # Chuẩn bị đặc trưng và mục tiêu
    features = config['features']
    target = 'ETV'
    for col in features:
        if col in ['Team', 'Nation']:
            df_filtered[col] = df_filtered[col].fillna('Unknown')
        else:
            df_filtered[col] = pd.to_numeric(df_filtered[col], errors='coerce')
            df_filtered[col] = df_filtered[col].fillna(df_filtered[col].median()
if not pd.isna(df_filtered[col].median()) else 0)
            df_filtered[col] = np.log1p(df_filtered[col].clip(lower=0))

    df_ml = df_filtered.dropna(subset=[target]).copy()
    if df_ml.empty:
        print(f"Lỗi: Không có dữ liệu ETV hợp lệ cho {position}.")
        return None, unmatched

    X = df_ml[features]
    y = df_ml[target]
```

- **Ý nghĩa**:
  - Đọc file results.csv và player_transfer_fee.csv.
  - Lọc cầu thủ theo vị trí chính (Primary_Position).
  - So khớp tên cầu thủ, gán giá trị ETV từ player_transfer_fee.csv sau khi chuyển đổi bằng parse_etv.
  - Chuẩn bị dữ liệu:
    - Điền Unknown cho Team, Nation nếu thiếu.
    - Chuyển đổi cột số, điền giá trị thiếu bằng trung vị, áp dụng log để giảm lệch.

- Tạo tập đặc trưng (X) và mục tiêu (y), trả về danh sách cầu thủ không khớp.
- **Mục đích**: Chuẩn bị dữ liệu sạch, khớp tên cầu thủ, và sẵn sàng huấn luyện mô hình.

6. Huấn luyện và đánh giá mô hình

```
    # Chia dữ liệu
    if len(df_ml) > 5:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    else:
        print(f"Cảnh báo: Không đủ dữ liệu cho {position} để chia train/test.")
        X_train, y_train = X, y
        X_test, y_test = X, y

    # Tạo pipeline
    numeric_features = [col for col in features if col not in ['Team', 'Nation']]
    categorical_features = [col for col in features if col in ['Team', 'Nation']]
    pipeline = Pipeline([
        ('encoder', TargetEncoder(cols=categorical_features)),  # Mã hóa Team,
Nation
        ('scaler', StandardScaler()),  # Chuẩn hóa dữ liệu số
        ('model', GradientBoostingRegressor(n_estimators=100, max_depth=3,
learning_rate=0.1, random_state=42))
    ])

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

    # Đánh giá mô hình
    if len(X_test) > 0:
        y_pred = pipeline.predict(X_test)
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))
        r2 = r2_score(y_test, y_pred)
        print(f"Đánh giá cho {position}: RMSE = {rmse:.2f}, R² = {r2:.2f}")

    # Dự đoán ETV
    df_filtered['Predicted_Transfer_Value'] =
pipeline.predict(df_filtered[features])
    df_filtered['Predicted_Transfer_Value'] =
df_filtered['Predicted_Transfer_Value'].clip(lower=100_000, upper=200_000_000)
    df_filtered['Predicted_Transfer_Value_M'] =
(df_filtered['Predicted_Transfer_Value'] / 1_000_000).round(2)
    df_filtered['Actual_Transfer_Value_M'] = (df_filtered['ETV'] /
1_000_000).round(2)
```

```
    # Chuẩn bị đầu ra
    output_columns = ['Player', 'Team', 'Nation', 'Position',
'Actual_Transfer_Value_M', 'Predicted_Transfer_Value_M']
    df_filtered['Position'] = position
    result = df_filtered[output_columns].copy()

    return result, unmatched
```

- **Ý nghĩa**:
  - Chia dữ liệu thành train/test (80/20) nếu đủ mẫu (>5), nếu không dùng toàn bộ dữ liệu.
  - Tạo pipeline: mã hóa Team, Nation bằng TargetEncoder, chuẩn hóa số bằng StandardScaler, huấn luyện bằng GradientBoostingRegressor.
  - Đánh giá mô hình bằng RMSE và R^2 trên tập test.
  - Dự đoán ETV, giới hạn giá trị trong khoảng 100,000 đến 200 triệu, chuyển thành triệu (M).
  - Chuẩn bị DataFrame kết quả với các cột cần thiết.
- **Mục đích**: Huấn luyện mô hình, dự đoán ETV, và chuẩn bị dữ liệu đầu ra cho mỗi vị trí.

7. Hàm chính và lưu kết quả

```
# Hàm chính
def main():
    """Chạy quy trình ước lượng giá trị cầu thủ cho tất cả vị trí."""
    csv_dir = get_csv_dir()
    results_path = os.path.join(csv_dir, "results.csv")  # Thay result.csv bằng
results.csv
    etv_path = os.path.join(csv_dir, "player_transfer_fee.csv")
    output_path = os.path.join(csv_dir, "ml_transfer_values_gradient.csv")

    all_results = []
    all_unmatched = []

    for position, config in positions_config.items():
        print(f"\nXử lý vị trí {position}...")
        result, unmatched = process_position(position, config, results_path,
etv_path)
        if result is not None:
            all_results.append(result)
        if unmatched:
            all_unmatched.extend([(position, player) for player in unmatched])

    if all_results:
        combined_results = pd.concat(all_results, ignore_index=True)
        combined_results =
combined_results.sort_values(by='Predicted_Transfer_Value_M', ascending=False)
```

```
        combined_results.to_csv(output_path, index=False)
        print(f"Kết quả đã được lưu vào: {output_path}")
    if all_unmatched:
        print(f"\nCầu thủ không khớp: {len(all_unmatched)}")
        for pos, player in all_unmatched:
            print(f"{pos}: {player}")

if __name__ == "__main__":
    main()
```

- **Ý nghĩa**:
  - Thiết lập đường dẫn cho các file: results.csv, player_transfer_fee.csv, ml_transfer_values_gradient.csv.
  - Xử lý từng vị trí (GK, DF, MF, FW), thu thập kết quả và danh sách cầu thủ không khớp.
  - Gộp kết quả, sắp xếp theo giá trị dự đoán (giảm dần), lưu vào ml_transfer_values_gradient.csv.
  - In danh sách cầu thủ không khớp (nếu có).
- **Mục đích**: Điều phối quy trình, gộp và lưu kết quả dự đoán ETV cho tất cả vị trí.

Đó là ý nghĩa và mục đích của mỗi phần code trong mỗi yêu cầu thầy giao .Cảm ơn thầy đã xem và đánh giá .