BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN

LẬP TRÌNH PYTHON

Giảng viên: **Kim Ngọc Bách**
Sinh viên thực hiện: **Dương Anh Đức**
Mã sinh viên: **B23DCCE018**
Lớp: **D23CQCE06-B**
Niên Khóa: **2024-2025**
Hệ đào tạo: **Đại học chính quy**

Hà Nội, Tháng 05/2025

Mục Lục

I. Bài 1 3

1. Nhập Thư Viện 3

2. Lớp Player 3

3. Các Phương Thức Cài Đặt của Lớp Player 7

4. Phương Thức str của Lớp Player 10

5. Lớp Player_Manager 10

6. Biến header_player và Hàm row_player 11

7. Lớp DataCrawler 13

8. Phương Thức extract_web_data của DataCrawler 13

9. Phương Thức process_standard_stats 14

11. Phương Thức export_to_csv 18

12. Phương Thức collect_all_data và Main 19

II. Bài 2 19

1. Nhập Thư Viện và Tải Dữ Liệu 19

2. Hàm find_extremes 20

3. Hàm calculate_stats 21

4. Hàm create_histograms 22

5. Hàm find_top_teams 23

6. Hàm analyze_team_performance 24

III. Bài 3 25

1. Nhập Thư Viện và Tải Dữ Liệu 25

2. Chuẩn Bị Dữ Liệu và Chuẩn Hóa 26

3. Phương Pháp Elbow để Chọn Số Cụm Tối Ưu 26

4. Phân Cụm với K-Means 28

5. Tóm Tắt Thống Kê Theo Cụm 29

6. Giảm Chiều Dữ Liệu và Trực Quan Hóa với PCA 29

7. Hàm find_extremes 31

8. Hàm calculate_stats 32

9. Hàm create_histograms 33

10. Hàm find_top_teams 34

11. Hàm analyze_team_performance 35

IV. Bài 4 36

1. Nhập Thư Viện và Chuẩn Bị Dữ Liệu 36

2. Khởi Tạo Trình Duyệt và Điều Hướng 37

3. Xử Lý Cookie và Đợi Bảng Dữ Liệu 37
4. Thu Thập Dữ Liệu Cầu Thủ Theo Từng Đội 38
5. Lưu Dữ Liệu và Đóng Trình Duyệt 40

I. Bài 1

1. Nhập Thư Viện

Mã Nguồn

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

import time
import csv
```

Giải Thích

- Mục đích: Nhập các thư viện cần thiết để thực hiện các chức năng thu thập dữ liệu web và xử lý tệp CSV.
- Chi tiết:
    - selenium.webdriver: Cung cấp công cụ để khởi tạo và điều khiển trình duyệt (ở đây là Chrome).
    - selenium.webdriver.common.by.By: Cho phép định vị các phần tử HTML bằng các phương thức như ID, XPath, hoặc tag name.
    - selenium.webdriver.support.ui.WebDriverWait: Hỗ trợ chờ cho đến khi một phần tử trên trang web xuất hiện, đảm bảo xử lý các trang tải động.
    - selenium.webdriver.support.expected_conditions (bí danh EC): Cung cấp các điều kiện chờ như sự hiện diện của phần tử theo XPath.
    - time: Được nhập nhưng không sử dụng trong mã, có thể dự phòng cho việc thêm độ trễ nếu cần.
    - csv: Dùng để ghi dữ liệu vào tệp CSV.

2. Lớp Player

Mã Nguồn

```
class Player:
    def __init__(self, name, nation, position, team, age):
        self.name = name
        self.nation = nation
        self.team = team
        self.position = position
        self.age = age
        self.playing_time = {
            "matches_played": "N/a",
            "starts": "N/a",
```

```
    "minutes": "N/a"
}
self.performance = {
    "goals": "N/a",
    "assists": "N/a",
    "yellow_cards": "N/a",
    "red_cards": "N/a"
}
self.expected = {
    "xG": "N/a",
    "xAG": "N/a"
}
self.progression = {
    "PrgC": "N/a",
    "PrgP": "N/a",
    "PrgR": "N/a"
}
self.per_90 = {
    "Gls": "N/a",
    "Ast": "N/a",
    "xG": "N/a",
    "xAG": "N/a",
}
self.goalkeeping = {
    "Performance": {
        "GA90": "N/a",
        "Save%": "N/a",
        "CS%": "N/a"
    },
    "Penalty Kicks": {
        "Save%": "N/a"
    }
}
self.shooting = {
    "SoT%": "N/a",
    "SoT/90": "N/a",
    "G/Sh": "N/a",
    "Dist": "N/a",
}
self.passing = {
    "Total": {
        "Cmp": "N/a",
        "Cmp%": "N/a",
        "TotDist": "N/a",
    },
    "Short": {
```

```
            "Cmp%": "N/a"
        },
        "Medium": {
            "Cmp%": "N/a"
        },
        "Long": {
            "Cmp%": "N/a"
        },
        "Expected": {
            "KP": "N/a",
            "1/3": "N/a",
            "PPA": "N/a",
            "CrsPA": "N/a",
            "PrgP": "N/a"
        }
    }
    self.goal_shot_creation = {
        "SCA": {
            "SCA": "N/a",
            "SCA90": "N/a"
        },
        "GCA": {
            "GCA": "N/a",
            "GCA90": "N/a"
        },
    }
    self.defensive_actions = {
        "Tackles": {
            "Tkl": "N/a",
            "TklW": "N/a",
        },
        "Challenges": {
            "Att": "N/a",
            "Lost": "N/a"
        },
        "Blocks": {
            "Blocks": "N/a",
            "Sh": "N/a",
            "Pass": "N/a",
            "Int": "N/a",
        }
    }
    self.possession = {
        "Touches": {
            "Touches": "N/a",
            "Def Pen": "N/a",
```

```
            "Def 3rd": "N/a",
            "Mid 3rd": "N/a",
            "Att 3rd": "N/a",
            "Att Pen": "N/a",
        },
        "Take-Ons": {
            "Att": "N/a",
            "Succ%": "N/a",
            "Tkld%": "N/a"
        },
        "Carries": {
            "Carries": "N/a",
            "ProDist": "N/a",
            "ProgC": "N/a",
            "1/3": "N/a",
            "CPA": "N/a",
            "Mis": "N/a",
            "Dis": "N/a"
        },
        "Receiving": {
            "Rec": "N/a",
            "PrgR": "N/a"
        }
    }
    self.misc_stats = {
        "Performance": {
            "Fls": "N/a",
            "Fld": "N/a",
            "Off": "N/a",
            "Crs": "N/a",
            "OG": "N/a",
            "Recov": "N/a"
        },
        "Aerial Duels": {
            "Won": "N/a",
            "Lost": "N/a",
            "Won%": "N/a"
        }
    }
```

Giải Thích

- Mục đích: Định nghĩa lớp Player để đại diện cho một cầu thủ bóng đá, lưu trữ thông tin cơ bản và các danh mục thống kê chi tiết.
- Chi tiết:
    - Hàm khởi tạo (__init__): Nhận các tham số name, nation, position, team, age và gán chúng vào các thuộc tính tương ứng.

- Danh mục thống kê: Khởi tạo các từ điển lồng nhau để lưu trữ dữ liệu thống kê, tất cả đều có giá trị mặc định là "N/a". Các danh mục bao gồm:
  - playing_time: Số trận đấu, số trận đá chính, số phút thi đấu.
  - performance: Bàn thắng, kiến tạo, thẻ vàng, thẻ đỏ.
  - expected: Bàn thắng kỳ vọng (xG), kiến tạo kỳ vọng (xAG).
  - progression: Mang bóng tiến lên, chuyền bóng tiến lên, nhận bóng tiến lên.
  - per_90: Chỉ số mỗi 90 phút (bàn thắng, kiến tạo, xG, xAG).
  - goalkeeping: Thống kê thủ môn như bàn thua mỗi 90 phút, tỷ lệ cứu thua.
  - shooting: Tỷ lệ sút trúng đích, bàn thắng mỗi cú sút, khoảng cách sút.
  - passing: Thống kê chuyền bóng tổng quát, ngắn, trung bình, dài và các chỉ số kỳ vọng như đường chuyền then chốt (KP).
  - goal_shot_creation: Hành động tạo cơ hội sút (SCA) và tạo bàn thắng (GCA).
  - defensive_actions: Tắc bóng, tranh chấp, chặn bóng, đánh chặn.
  - possession: Chạm bóng, rê bóng, mang bóng, nhận bóng.
  - misc_stats: Phạt lỗi, việt vị, thu hồi bóng, tranh chấp trên không.
- Ứng dụng: Lớp này cung cấp cấu trúc dữ liệu linh hoạt, cho phép lưu trữ và truy cập thống kê cầu thủ một cách có tổ chức.

3. Các Phương Thức Cài Đặt của Lớp Player

Mã Nguồn

```
  def setPlaying_time(self, arr):
    self.playing_time["matches_played"] = arr[0]
    self.playing_time["starts"] = arr[1]
    self.playing_time["minutes"] = arr[2]

  def setPerformance(self, arr):
    self.performance["goals"] = arr[0]
    self.performance["assists"] = arr[2]
    self.performance["yellow_cards"] = arr[3]
    self.performance["red_cards"] = arr[4]

  def setExpected(self, arr):
    self.expected["xG"] = arr[0]
    self.expected["xAG"] = arr[2]

  def setProgression(self, arr):
    self.progression["PrgC"] = arr[0]
    self.progression["PrgP"] = arr[1]
    self.progression["PrgR"] = arr[2]

  def setPer90(self, arr):
    self.per_90["Gls"] = arr[0]
    self.per_90["Ast"] = arr[1]
```

```
        self.per_90["xG"] = arr[2]
        self.per_90["xAG"] = arr[3]

    def setGoalkeeping(self, performance_arr, penalty_arr):
        self.goalkeeping["Performance"]["GA90"] = performance_arr[1]
        self.goalkeeping["Performance"]["Save%"] = performance_arr[4]
        self.goalkeeping["Performance"]["CS%"] = performance_arr[9]
        self.goalkeeping["Penalty Kicks"]["Save%"] = penalty_arr[4]

    def setShooting(self, arr):
        self.shooting["SoT%"] = arr[0]
        self.shooting["SoT/90"] = arr[1]
        self.shooting["G/Sh"] = arr[2]
        self.shooting["Dist"] = arr[3]

    def setPassing(self, total_arr, short_arr, medium_arr, long_arr, expected_arr):
        self.passing["Total"]["Cmp"] = total_arr[0]
        self.passing["Total"]["Cmp%"] = total_arr[2]
        self.passing["Total"]["TotDist"] = total_arr[3]
        self.passing["Short"]["Cmp%"] = short_arr[2]
        self.passing["Medium"]["Cmp%"] = medium_arr[2]
        self.passing["Long"]["Cmp%"] = long_arr[2]
        self.passing["Expected"]["KP"] = expected_arr[4]
        self.passing["Expected"]["1/3"] = expected_arr[5]
        self.passing["Expected"]["PPA"] = expected_arr[6]
        self.passing["Expected"]["CrsPA"] = expected_arr[7]
        self.passing["Expected"]["PrgP"] = expected_arr[8]

    def setGoalShotCreation(self, sca_arr, gca_arr):
        self.goal_shot_creation["SCA"]["SCA"] = sca_arr[0]
        self.goal_shot_creation["SCA"]["SCA90"] = sca_arr[1]
        self.goal_shot_creation["GCA"]["GCA"] = gca_arr[0]
        self.goal_shot_creation["GCA"]["GCA90"] = gca_arr[1]

    def setDefensiveActions(self, tackles_arr, challenges_arr, blocks_arr):
        self.defensive_actions["Tackles"]["Tkl"] = tackles_arr[0]
        self.defensive_actions["Tackles"]["TklW"] = tackles_arr[1]
        self.defensive_actions["Challenges"]["Att"] = challenges_arr[1]
        self.defensive_actions["Challenges"]["Lost"] = challenges_arr[3]
        self.defensive_actions["Blocks"]["Blocks"] = blocks_arr[0]
        self.defensive_actions["Blocks"]["Sh"] = blocks_arr[1]
        self.defensive_actions["Blocks"]["Pass"] = blocks_arr[2]
        self.defensive_actions["Blocks"]["Int"] = blocks_arr[3]

    def setPossession(self, touches_arr, take_ons_arr, carries_arr, receiving_arr):
        self.possession["Touches"]["Touches"] = touches_arr[0]
```

```
        self.possession["Touches"]["Def Pen"] = touches_arr[1]
        self.possession["Touches"]["Def 3rd"] = touches_arr[2]
        self.possession["Touches"]["Mid 3rd"] = touches_arr[3]
        self.possession["Touches"]["Att 3rd"] = touches_arr[4]
        self.possession["Touches"]["Att Pen"] = touches_arr[5]
        self.possession["Take-Ons"]["Att"] = take_ons_arr[0]
        self.possession["Take-Ons"]["Succ%"] = take_ons_arr[2]
        self.possession["Take-Ons"]["Tkld%"] = take_ons_arr[4]
        self.possession["Carries"]["Carries"] = carries_arr[0]
        self.possession["Carries"]["ProDist"] = carries_arr[2]
        self.possession["Carries"]["ProgC"] = carries_arr[3]
        self.possession["Carries"]["1/3"] = carries_arr[4]
        self.possession["Carries"]["CPA"] = carries_arr[5]
        self.possession["Carries"]["Mis"] = carries_arr[6]
        self.possession["Carries"]["Dis"] = carries_arr[7]
        self.possession["Receiving"]["Rec"] = receiving_arr[0]
        self.possession["Receiving"]["PrgR"] = receiving_arr[1]

    def setMiscStats(self, performance_arr, aerial_duels_arr):
        self.misc_stats["Performance"]["Fls"] = performance_arr[0]
        self.misc_stats["Performance"]["Fld"] = performance_arr[1]
        self.misc_stats["Performance"]["Off"] = performance_arr[2]
        self.misc_stats["Performance"]["Crs"] = performance_arr[3]
        self.misc_stats["Performance"]["OG"] = performance_arr[4]
        self.misc_stats["Performance"]["Recov"] = performance_arr[5]
        self.misc_stats["Aerial Duels"]["Won"] = aerial_duels_arr[0]
        self.misc_stats["Aerial Duels"]["Lost"] = aerial_duels_arr[1]
        self.misc_stats["Aerial Duels"]["Won%"] = aerial_duels_arr[2]
```

Giải Thích

- Mục đích: Cung cấp các phương thức để cập nhật dữ liệu thống kê của cầu thủ từ các mảng dữ liệu trích xuất từ trang web.
- Chi tiết:
    - setPlaying_time(arr): Cập nhật số trận đấu, số trận đá chính và số phút thi đấu từ mảng 3 phần tử.
    - setPerformance(arr): Cập nhật bàn thắng, kiến tạo, thẻ vàng và thẻ đỏ từ mảng 5 phần tử (lưu ý thứ tự không liên tục).
    - setExpected(arr): Cập nhật xG và xAG từ mảng 3 phần tử (bỏ qua phần tử giữa).
    - setProgression(arr): Cập nhật các chỉ số tiến lên (PrgC, PrgP, PrgR) từ mảng 3 phần tử.
    - setPer90(arr): Cập nhật chỉ số mỗi 90 phút (bàn thắng, kiến tạo, xG, xAG) từ mảng 4 phần tử.
    - setGoalkeeping(performance_arr, penalty_arr): Cập nhật thống kê thủ môn từ hai mảng, bao gồm bàn thua mỗi 90 phút, tỷ lệ cứu thua, tỷ lệ giữ sạch lưới và tỷ lệ cứu phạt đền.
    - setShooting(arr): Cập nhật thống kê sút bóng từ mảng 4 phần tử.

    - setPassing(total_arr, short_arr, medium_arr, long_arr, expected_arr): Cập nhật thống kê chuyền bóng từ nhiều mảng, bao gồm số đường chuyền hoàn thành, tỷ lệ hoàn thành, khoảng cách chuyền và các chỉ số kỳ vọng.
    - setGoalShotCreation(sca_arr, gca_arr): Cập nhật hành động tạo cơ hội sút và tạo bàn thắng từ hai mảng.
    - setDefensiveActions(tackles_arr, challenges_arr, blocks_arr): Cập nhật hành động phòng ngự từ ba mảng, bao gồm tắc bóng, tranh chấp và chặn bóng.
    - setPossession(touches_arr, take_ons_arr, carries_arr, receiving_arr): Cập nhật thống kê kiểm soát bóng từ bốn mảng, bao gồm chạm bóng, rê bóng, mang bóng và nhận bóng.
    - setMiscStats(performance_arr, aerial_duels_arr): Cập nhật thống kê khác từ hai mảng, bao gồm phạt lỗi, việt vị, thu hồi bóng và tranh chấp trên không.
- Ứng dụng: Các phương thức này đảm bảo dữ liệu được gán đúng vào cấu trúc từ điển của cầu thủ, cho phép xử lý dữ liệu từ các nguồn khác nhau một cách nhất quán.

4. Phương Thức str của Lớp Player

Mã Nguồn

```
  def __str__(self) -> str:
    return self.name + " " + str(self.age) + " " + self.team + "\n" + str(self.performance) + \
      "\n" + str(self.per_90) + \
      "\n" + str(self.goalkeeping) + \
      "\n" + str(self.shooting)
```

Giải Thích

- Mục đích: Cung cấp biểu diễn chuỗi của đối tượng Player để hiển thị thông tin cơ bản và một số danh mục thống kê.
- Chi tiết:
    - Trả về một chuỗi gồm tên, tuổi, đội bóng, và các từ điển performance, per_90, goalkeeping, shooting, mỗi mục trên một dòng.
    - Dùng để gỡ lỗi hoặc in thông tin cầu thủ một cách dễ đọc.
- Ứng dụng: Hữu ích khi cần kiểm tra nhanh dữ liệu của một cầu thủ, nhưng không hiển thị toàn bộ thống kê.

5. Lớp Player_Manager

Mã Nguồn

```
class Player_Manager:
  def __init__(self) -> None:
    self.list_player = []

  def add(self, player):
    self.list_player.append(player)

  def find(self, name, team):
    for i in self.list_player:
      if i.name == name and i.team == team:
        return i
    return None
```

```
    def filtering(self):
        self.list_player = list(filter(lambda p: p.playing_time["minutes"] > 90, self.list_player))

    def show(self):
        for i in self.list_player:
            print(i)

    def sort(self):
        self.list_player = sorted(self.list_player, key=lambda x: (x.name.split()[-1], -x.age))
```

Giải Thích

- Mục đích: Quản lý danh sách các đối tượng Player và cung cấp các phương thức để thao tác với danh sách.
- Chi tiết:
    - init: Khởi tạo một danh sách rỗng list_player để lưu trữ các cầu thủ.
    - add(player): Thêm một đối tượng Player vào danh sách.
    - find(name, team): Tìm kiếm cầu thủ theo tên và đội bóng, trả về đối tượng Player nếu tìm thấy, hoặc None nếu không.
    - filtering(): Lọc danh sách, chỉ giữ lại các cầu thủ có số phút thi đấu lớn hơn 90 phút (loại bỏ các cầu thủ ít ra sân).
    - show(): In thông tin tất cả cầu thủ trong danh sách (dùng __str__ của Player), chủ yếu để gỡ lỗi.
    - sort(): Sắp xếp danh sách cầu thủ theo họ (lấy từ phần tử cuối của tên bằng name.split()[-1]) và sau đó theo tuổi giảm dần.
- Ứng dụng: Lớp này cung cấp giao diện để quản lý tập hợp cầu thủ, đảm bảo các thao tác như thêm, tìm kiếm, lọc và sắp xếp được thực hiện dễ dàng.

6. Biến header_player và Hàm row_player

Mã Nguồn

```
header_player = [
    "name", "nation", "team", "position", "age",
    "matches_played", "starts", "minutes",
    "goals", "assists", "yellow_cards", "red_cards",
    "xG", "xAG",
    "PrgC", "PrgP", "PrgR",
    "per90_Gls", "per90_Ast", "per90_xG", "per90_xAG",
    "SoT%", "SoT/90", "G/Sh", "Dist",
    "Pass_Cmp", "Pass_Cmp%", "TotDist",
    "Short_Cmp%", "Medium_Cmp%", "Long_Cmp%",
    "KP", "1/3", "PPA", "CrsPA", "PrgP",
    "Tkl", "TklW", "Challenges_Att", "Challenges_Lost",
    "Blocks", "Blocks_SH", "Blocks_Pass", "Blocks_Int",
    "Touches", "Def_Pen", "Def_3rd", "Mid_3rd", "Att_3rd", "Att_Pen",
    "Take_Att", "Take_Succ%", "Take_Tkld%",
    "Carries", "Carries_ProDist", "Carries_ProgC", "Carries_1/3", "Carries_CPA",
"Carries_Mis", "Carries_Dis",
```

```
    "REC", "REC_PrgR",
    "Fls", "Fld", "Off", "Crs", "OG", "Recov", "Aerial_Won", "Aerial_Lost", "Aerial_Won%"
]

def row_player(player):
    return [
        player.name, player.nation, player.team, player.position, player.age,
        player.playing_time["matches_played"], player.playing_time["starts"],
player.playing_time["minutes"],
        player.performance["goals"], player.performance["assists"],
player.performance["yellow_cards"], player.performance["red_cards"],
        player.expected["xG"], player.expected["xAG"],
        player.progression["PrgC"], player.progression["PrgP"], player.progression["PrgR"],
        player.per_90["Gls"], player.per_90["Ast"], player.per_90["xG"], player.per_90["xAG"],
        player.shooting["SoT%"], player.shooting["SoT/90"], player.shooting["G/Sh"],
player.shooting["Dist"],
        player.passing["Total"]["Cmp"], player.passing["Total"]["Cmp%"],
player.passing["Total"]["TotDist"],
        player.passing["Short"]["Cmp%"], player.passing["Medium"]["Cmp%"],
player.passing["Long"]["Cmp%"],
        player.passing["Expected"]["KP"], player.passing["Expected"]["1/3"],
player.passing["Expected"]["PPA"],
        player.passing["Expected"]["CrsPA"], player.passing["Expected"]["PrgP"],
        player.defensive_actions["Tackles"]["Tkl"],
player.defensive_actions["Tackles"]["TklW"],
        player.defensive_actions["Challenges"]["Att"],
player.defensive_actions["Challenges"]["Lost"],
        player.defensive_actions["Blocks"]["Blocks"],
player.defensive_actions["Blocks"]["Sh"],
        player.defensive_actions["Blocks"]["Pass"], player.defensive_actions["Blocks"]["Int"],
        player.possession["Touches"]["Touches"], player.possession["Touches"]["Def Pen"],
        player.possession["Touches"]["Def 3rd"], player.possession["Touches"]["Mid 3rd"],
        player.possession["Touches"]["Att 3rd"], player.possession["Touches"]["Att Pen"],
        player.possession["Take-Ons"]["Att"], player.possession["Take-Ons"]["Succ%"],
        player.possession["Take-Ons"]["Tkld%"],
        player.possession["Carries"]["Carries"], player.possession["Carries"]["ProDist"],
        player.possession["Carries"]["ProgC"], player.possession["Carries"]["1/3"],
        player.possession["Carries"]["CPA"], player.possession["Carries"]["Mis"],
        player.possession["Carries"]["Dis"],
        player.possession["Receiving"]["Rec"], player.possession["Receiving"]["PrgR"],
        player.misc_stats["Performance"]["Fls"], player.misc_stats["Performance"]["Fld"],
        player.misc_stats["Performance"]["Off"], player.misc_stats["Performance"]["Crs"],
        player.misc_stats["Performance"]["OG"], player.misc_stats["Performance"]["Recov"],
        player.misc_stats["Aerial Duels"]["Won"], player.misc_stats["Aerial Duels"]["Lost"],
        player.misc_stats["Aerial Duels"]["Won%"]
    ]
```

Giải Thích

- Mục đích: Cung cấp tiêu đề cột và hàm để chuyển đổi dữ liệu cầu thủ thành hàng CSV.
- Chi tiết:
  - header_player: Danh sách các tiêu đề cột cho tệp CSV, bao gồm tất cả các trường từ thông tin cơ bản (tên, tuổi) đến thống kê chi tiết (tắc bóng, chạm bóng, v.v.).
  - row_player(player): Trả về một danh sách chứa tất cả dữ liệu của một cầu thủ, theo đúng thứ tự của header_player. Hàm này truy cập các thuộc tính và từ điển lồng nhau của đối tượng Player để tạo hàng CSV.
- Ứng dụng: Đảm bảo dữ liệu được xuất ra CSV có cấu trúc nhất quán và đầy đủ.

7. Lớp DataCrawler

Mã Nguồn

```
class DataCrawler:
    def __init__(self):
        self.player_manager = Player_Manager()

    def convert_to_float(self, value):
        return float(value) if value != '' else "N/a"
```

Giải Thích

- Mục đích: Khởi tạo lớp DataCrawler để thu thập, xử lý và xuất dữ liệu, cùng với hàm hỗ trợ chuyển đổi dữ liệu.
- Chi tiết:
  - init: Tạo một thể hiện Player_Manager để quản lý danh sách cầu thủ.
  - convert_to_float(value): Chuyển đổi một chuỗi thành số thực, trả về "N/a" nếu chuỗi rỗng. Hàm này dùng để xử lý các giá trị số từ trang web.
- Ứng dụng: Lớp này là trung tâm điều phối việc thu thập và xử lý dữ liệu, trong khi convert_to_float đảm bảo dữ liệu số được xử lý đúng cách.

8. Phương Thức extract_web_data của DataCrawler

Mã Nguồn

```
    def extract_web_data(self, url, xpath, data_category):
        driver = webdriver.Chrome()
        extracted_data = []

        try:
            driver.get(url)
            table = WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.XPATH, xpath))
            )
            all_rows = table.find_elements(By.TAG_NAME, 'tr')

            for row in all_rows:
                columns = row.find_elements(By.TAG_NAME, 'td')
                row_data = []
```

```
            for col_index, column in enumerate(columns[:-1]):
                text_content = column.text.strip()

                if col_index == 1:
                    words = text_content.split()
                    row_data.append(words[1] if len(words) == 2 else text_content)
                else:
                    if col_index >= 4:
                        text_content = text_content.split("-")[0]
                        text_content = text_content.replace(",", "")
                        text_content = self.convert_to_float(text_content)
                    row_data.append(text_content)

            if row_data:
                extracted_data.append(row_data)
        finally:
            driver.quit()
            print(f"Completed crawling {data_category} data")

        return extracted_data
```

Giải Thích

- Mục đích: Trích xuất dữ liệu từ một bảng HTML trên trang web theo URL và XPath được cung cấp.
- Chi tiết:
    - Khởi tạo WebDriver Chrome và danh sách extracted_data để lưu dữ liệu.
    - Điều hướng đến url và chờ tối đa 10 giây cho đến khi bảng được định vị bởi xpath xuất hiện.
    - Lấy tất cả các hàng (tr) từ bảng và duyệt qua từng hàng.
    - Đối với mỗi hàng, lấy các cột (td) và xử lý nội dung văn bản:
        - Cột thứ hai (quốc tịch): Tách chuỗi và lấy từ thứ hai nếu có (ví dụ: "ENG England" → "England").
        - Các cột số (chỉ số ≥ 4): Loại bỏ hậu tố (ví dụ: "1-0" → "1"), dấu phẩy (ví dụ: "1,234" → "1234"), và chuyển thành số thực bằng convert_to_float.
        - Các cột khác: Lưu nguyên văn bản.
    - Thêm hàng dữ liệu vào extracted_data nếu không rỗng.
    - Đóng trình duyệt bằng driver.quit() trong khối finally để đảm bảo giải phóng tài nguyên.
    - In thông báo hoàn thành với data_category (ví dụ: "Standard", "Goalkeeping").
    - Trả về danh sách các hàng dữ liệu đã xử lý.
- Ứng dụng: Hàm này là cốt lõi của quá trình thu thập dữ liệu, được sử dụng bởi tất cả các phương thức xử lý thống kê.

9. Phương Thức process_standard_stats

Mã Nguồn

```
    def process_standard_stats(self):
        url = "https://fbref.com/en/comps/9/2024-2025/stats/2024-2025-Premier-League-Stats"
        data = self.extract_web_data(url, '//*[@id="stats_standard"]', "Standard")
        print("Standard data sample:", data[0])

        for player_data in data:
            player = self.player_manager.find(player_data[0], player_data[3])

            if not player:
                player = Player(
                    player_data[0],
                    player_data[1],
                    player_data[2],
                    player_data[3],
                    player_data[4]
                )

                player.setPlaying_time(player_data[6:9])
                player.setPerformance([
                    player_data[13],
                    player_data[14],
                    player_data[11],
                    player_data[16],
                    player_data[17]
                ])
                player.setExpected(player_data[18:21])
                player.setProgression(player_data[22:25])
                player.setPer90(player_data[25:])

                self.player_manager.add(player)

        self.player_manager.filtering()
```

Giải Thích

- Mục đích: Thu thập và xử lý thống kê cơ bản (Standard Stats) từ trang FBref, tạo các đối tượng Player và lọc cầu thủ.
- Chi tiết:
    - Gọi extract_web_data với URL và XPath của bảng thống kê cơ bản.
    - In mẫu dữ liệu đầu tiên để kiểm tra.
    - Duyệt qua mỗi hàng dữ liệu (player_data):
        - Tìm cầu thủ trong player_manager bằng tên (player_data[0]) và đội bóng (player_data[3]).
        - Nếu không tìm thấy, tạo đối tượng Player mới với thông tin cơ bản (tên, quốc tịch, vị trí, đội, tuổi).
        - Cập nhật thống kê:
            - setPlaying_time: Dùng player_data[6:9] (số trận, đá chính, phút).

        - setPerformance: Dùng các chỉ số không liên tục từ player_data (bàn thắng, kiến tạo, thẻ vàng, thẻ đỏ).
        - setExpected: Dùng player_data[18:21] (xG, xAG).
        - setProgression: Dùng player_data[22:25] (PrgC, PrgP, PrgR).
        - setPer90: Dùng player_data[25:] (chỉ số mỗi 90 phút).
      - Thêm cầu thủ vào player_manager.
    - Gọi player_manager.filtering() để loại bỏ cầu thủ có số phút thi đấu ≤ 90.
- Ứng dụng: Đây là phương thức chính để khởi tạo danh sách cầu thủ với thống kê cơ bản, làm nền tảng cho các phương thức xử lý khác.

10. Các Phương Thức Xử Lý Thống Kê Khác

Mã Nguồn

```
  def process_goalkeeper_stats(self):
    url = 'https://fbref.com/en/comps/9/2024-2025/keepers/2024-2025-Premier-League-Stats'
    data = self.extract_web_data(url, '//*[@id="stats_keeper"]', "Goalkeeping")

    for player_data in data:
      player = self.player_manager.find(player_data[0], player_data[3])
      if player:
        player.setGoalkeeping(player_data[10:20], player_data[20:])

  def process_shooting_stats(self):
    url = 'https://fbref.com/en/comps/9/2024-2025/shooting/2024-2025-Premier-League-
Stats'
    data = self.extract_web_data(url, '//*[@id="stats_shooting"]', "Shooting")

    for player_data in data:
      player = self.player_manager.find(player_data[0], player_data[3])
      if player:
        player.setShooting(player_data[7:19])

  def process_passing_stats(self):
    url = 'https://fbref.com/en/comps/9/2024-2025/passing/2024-2025-Premier-League-Stats'
    data = self.extract_web_data(url, '//*[@id="stats_passing"]', "Passing")

    for player_data in data:
      player = self.player_manager.find(player_data[0], player_data[3])
      if player:
        player.setPassing(
          player_data[7:12],
          player_data[12:15],
          player_data[15:18],
          player_data[18:21],
          player_data[21:]
        )

  def process_goal_creation_stats(self):
```

```
        url = 'https://fbref.com/en/comps/9/2024-2025/gca/2024-2025-Premier-League-Stats'
        data = self.extract_web_data(url, '//*[@id="stats_gca"]', "Goal and Shot Creation")

        for player_data in data:
            player = self.player_manager.find(player_data[0], player_data[3])
            if player:
                player.setGoalShotCreation(player_data[7:9], player_data[15:17])

    def process_defensive_stats(self):
        url = 'https://fbref.com/en/comps/9/2024-2025/defense/2024-2025-Premier-League-Stats'
        data = self.extract_web_data(url, '//*[@id="stats_defense"]', "Defensive Actions")

        for player_data in data:
            player = self.player_manager.find(player_data[0], player_data[3])
            if player:
                player.setDefensiveActions(
                    player_data[7:12],
                    player_data[12:16],
                    player_data[16:23]
                )

    def process_possession_stats(self):
        url = 'https://fbref.com/en/comps/9/2024-2025/possession/2024-2025-Premier-League-
Stats'
        data = self.extract_web_data(url, '//*[@id="stats_possession"]', "Possession")

        for player_data in data:
            player = self.player_manager.find(player_data[0], player_data[3])
            if player:
                player.setPossession(
                    player_data[7:14],
                    player_data[14:19],
                    player_data[19:27],
                    player_data[27:29]
                )

    def process_miscellaneous_stats(self):
        url = 'https://fbref.com/en/comps/9/2024-2025/misc/2024-2025-Premier-League-Stats'
        data = self.extract_web_data(url, '//*[@id="stats_misc"]', "Miscellaneous")

        for player_data in data:
            player = self.player_manager.find(player_data[0], player_data[3])
            if player:
                misc_data = player_data[10:14] + player_data[18:20]
                player.setMiscStats(misc_data, player_data[20:23])
```

Giải Thích

- Mục đích: Thu thập và cập nhật các danh mục thống kê bổ sung cho các cầu thủ đã có trong player_manager.
- Chi tiết:
    - Mỗi phương thức xử lý một danh mục thống kê cụ thể (thủ môn, sút bóng, chuyền bóng, v.v.) từ trang FBref tương ứng.
    - Quy trình chung:
        - Gọi extract_web_data với URL và XPath của bảng thống kê.
        - Duyệt qua mỗi hàng dữ liệu, tìm cầu thủ bằng player_manager.find.
        - Nếu tìm thấy cầu thủ, cập nhật thống kê bằng phương thức cài đặt tương ứng với các đoạn dữ liệu được cắt từ hàng.
    - process_goalkeeper_stats: Cập nhật thống kê thủ môn (bàn thua, cứu thua, giữ sạch lưới).
    - process_shooting_stats: Cập nhật thống kê sút bóng (sút trúng đích, khoảng cách sút).
    - process_passing_stats: Cập nhật thống kê chuyền bóng (chuyền ngắn, dài, đường chuyền then chốt).
    - process_goal_creation_stats: Cập nhật hành động tạo cơ hội sút và tạo bàn thắng.
    - process_defensive_stats: Cập nhật hành động phòng ngự (tắc bóng, chặn bóng).
    - process_possession_stats: Cập nhật thống kê kiểm soát bóng (chạm bóng, rê bóng, mang bóng).
    - process_miscellaneous_stats: Cập nhật thống kê khác (phạt lỗi, thu hồi bóng, tranh chấp trên không).
- Ứng dụng: Các phương thức này cho phép mở rộng dữ liệu cầu thủ với các thống kê chi tiết, nhưng hiện không được gọi trong luồng chính (collect_all_data).

11. Phương Thức export_to_csv

Mã Nguồn

```
  def export_to_csv(self, filename='Bài 1/result.csv'):
    with open(filename, mode='w', newline='', encoding='utf-8') as file:
      writer = csv.writer(file)

      writer.writerow(header_player)

      for player in self.player_manager.list_player:
        writer.writerow(row_player(player))

    print(f"Data exported to {filename}")
```

Giải Thích

- Mục đích: Xuất danh sách cầu thủ và thống kê của họ ra tệp CSV.
- Chi tiết:
    - Mở tệp CSV (mặc định: Bài 1/result.csv) ở chế độ ghi với mã hóa UTF-8.
    - Tạo đối tượng csv.writer để ghi dữ liệu.
    - Ghi hàng tiêu đề từ header_player.

    - Duyệt qua danh sách cầu thủ trong player_manager, gọi row_player để tạo hàng dữ liệu và ghi vào tệp.
    - In thông báo xác nhận khi hoàn thành.
- Ứng dụng: Đảm bảo dữ liệu được lưu trữ ở định dạng CSV, dễ dàng sử dụng cho phân tích hoặc nhập vào các công cụ khác.

12. Phương Thức collect_all_data và Main

Mã Nguồn

```
  def collect_all_data(self):
    self.process_standard_stats()
    self.player_manager.sort()
    self.export_to_csv()

if __name__ == "__main__":
  crawler = DataCrawler()
  crawler.collect_all_data()
```

Giải Thích

- Mục đích: Điều phối toàn bộ quá trình thu thập và xuất dữ liệu, đồng thời khởi chạy chương trình.
- Chi tiết:
    - collect_all_data:
        - Gọi process_standard_stats để thu thập thống kê cơ bản và tạo danh sách cầu thủ.
        - Gọi player_manager.sort để sắp xếp cầu thủ theo họ và tuổi.
        - Gọi export_to_csv để xuất dữ liệu ra tệp CSV.
    - Main block:
        - Tạo thể hiện DataCrawler và gọi collect_all_data khi mã được chạy trực tiếp.
- Ứng dụng: Đây là điểm nhập chính của chương trình, đảm bảo các bước được thực hiện theo thứ tự hợp lý.

II. Bài 2

1. Nhập Thư Viện và Tải Dữ Liệu

Mã Nguồn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

plt.ioff()

df = pd.read_csv('Bài 1/result.csv')

numeric_columns = [
  'age', 'matches_played', 'starts', 'minutes', 'goals', 'assists',
  'yellow_cards', 'red_cards', 'xG', 'xAG', 'PrgC', 'PrgP', 'PrgR',
```

```
    'per90_Gls', 'per90_Ast', 'per90_xG', 'per90_xAG', 'SoT%', 'SoT/90',
    'G/Sh', 'Dist', 'Pass_Cmp', 'Pass_Cmp%', 'TotDist', 'Short_Cmp%',
    'Medium_Cmp%', 'Long_Cmp%', 'KP', '1/3', 'PPA', 'CrsPA', 'SCA', 'SCA90',
    'GCA', 'GCA90', 'Tkl', 'TklW', 'Challenges_Att', 'Challenges_Lost',
    'Blocks', 'Blocks_SH', 'Blocks_Pass', 'Blocks_Int', 'Touches', 'Def_Pen',
    'Def_3rd', 'Mid_3rd', 'Att_3rd', 'Att_Pen', 'Take_Att', 'Take_Succ%',
    'Take_Tkld%', 'Carries', 'Carries_ProDist', 'Carries_ProgC', 'Carries_1/3',
    'Carries_CPA', 'Carries_Mis', 'Carries_Dis', 'REC', 'REC_PrgR', 'Fls',
    'Fld', 'Off', 'Crs', 'OG', 'Recov', 'Aerial_Won', 'Aerial_Lost', 'Aerial_Won%'
]

for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

Giải Thích

- Mục đích: Nhập các thư viện cần thiết, tải dữ liệu từ tệp CSV, và chuẩn hóa các cột số để chuẩn bị cho phân tích.
- Chi tiết:
  - Nhập thư viện:
    - pandas (bí danh pd): Xử lý dữ liệu dạng bảng (DataFrame).
    - numpy (bí danh np): Hỗ trợ tính toán số học.
    - matplotlib.pyplot (bí danh plt): Tạo biểu đồ histogram.
    - os: Thao tác với hệ thống tệp, như tạo thư mục.
  - Tắt chế độ hiển thị tương tác: plt.ioff() ngăn Matplotlib hiển thị biểu đồ trực tiếp, chỉ lưu vào tệp.
  - Tải dữ liệu: pd.read_csv('Bài 1/result.csv') đọc dữ liệu cầu thủ vào DataFrame df.
  - Định nghĩa cột số: numeric_columns chứa danh sách các cột thống kê (hiệu suất, chuyền bóng, phòng ngự, v.v.).
  - Chuẩn hóa cột số: Vòng lặp sử dụng pd.to_numeric để chuyển đổi các cột thành kiểu số, với errors='coerce' thay giá trị không hợp lệ bằng NaN.
- Ứng dụng: Chuẩn bị dữ liệu số để phân tích, đảm bảo tính nhất quán cho các phép tính thống kê.

2. Hàm find_extremes

Mã Nguồn

```
def find_extremes(df, columns, output_file='Bài 2/top_3.txt'):
    os.makedirs('Bài 2', exist_ok=True)
    with open(output_file, 'w', encoding='utf-8') as f:
        for col in columns:
            if col not in df.columns or df[col].dropna().empty:
                f.write(f"\nStatistic: {col}\nNo valid data available\n")
                f.write("-" * 50 + "\n")
                continue
            top_players = df[['name', 'team', col]].dropna().sort_values(by=col,
ascending=False).head(3)
            bottom_players = df[['name', 'team', col]].dropna().sort_values(by=col,
ascending=True).head(3)
            f.write(f"\nStatistic: {col}\n")
            f.write("Top 3 Players:\n")
            for _, row in top_players.iterrows():
```

```
            f.write(f"{row['name']} ({row['team']}): {row[col]:.2f}\n")
        f.write("Bottom 3 Players:\n")
        for _, row in bottom_players.iterrows():
            f.write(f"{row['name']} ({row['team']}): {row[col]:.2f}\n")
        f.write("-" * 50 + "\n")

find_extremes(df, numeric_columns)
```

Giải Thích

- Mục đích: Xác định và ghi ra tệp văn bản 3 cầu thủ có giá trị cao nhất và thấp nhất cho mỗi chỉ số thống kê số.
- Chi tiết:
  - Tham số:
    - df: DataFrame chứa dữ liệu.
    - columns: Danh sách cột số.
    - output_file: Đường dẫn tệp đầu ra (mặc định: Bài 2/top_3.txt).
  - Tạo thư mục: os.makedirs('Bài 2', exist_ok=True) tạo thư mục Bài 2 nếu chưa tồn tại.
  - Xử lý dữ liệu:
    - Kiểm tra cột hợp lệ, nếu không có dữ liệu thì ghi thông báo "No valid data available".
    - top_players: Lấy 3 cầu thủ có giá trị cao nhất (sắp xếp giảm dần, lấy 3 hàng đầu sau khi loại bỏ NaN).
    - bottom_players: Lấy 3 cầu thủ có giá trị thấp nhất (sắp xếp tăng dần).
  - Ghi kết quả: Ghi tên, đội, và giá trị (làm tròn 2 chữ số thập phân) cho từng cầu thủ vào tệp.
  - Gọi hàm: Áp dụng cho df và numeric_columns.
- Ứng dụng: Cung cấp danh sách cầu thủ nổi bật hoặc kém nhất theo từng chỉ số, hỗ trợ phân tích hiệu suất cá nhân.

3. Hàm calculate_stats

Mã Nguồn

```
def calculate_stats(df, columns, output_file='Bài 2/results2.csv'):
    os.makedirs('Bài 2', exist_ok=True)
    teams = ['all (toàn bộ)'] + list(df['team'].unique())
    stats_data = {'Team': teams}
    for col in columns:
        if col not in df.columns or df[col].dropna().empty:
            continue
        stats_data[f'Trung vị của {col}'] = []
        stats_data[f'Trung bình của {col}'] = []
        stats_data[f'Độ lệch chuẩn của {col}'] = []
        median = df[col].median()
        mean = df[col].mean()
        std = df[col].std()
        stats_data[f'Trung vị của {col}'].append(median)
        stats_data[f'Trung bình của {col}'].append(mean)
        stats_data[f'Độ lệch chuẩn của {col}'].append(std)
        for team in teams[1:]:
            team_df = df[df['team'] == team]
            median = team_df[col].median()
```

```
            mean = team_df[col].mean()
            std = team_df[col].std()
            stats_data[f'Trung vị của {col}'].append(median)
            stats_data[f'Trung bình của {col}'].append(mean)
            stats_data[f'Độ lệch chuẩn của {col}'].append(std)
    stats_df = pd.DataFrame(stats_data)
    stats_df.to_csv(output_file, index=False, encoding='utf-8')

calculate_stats(df, numeric_columns)
```

Giải Thích

- **Mục đích**: Tính toán trung vị, trung bình, và độ lệch chuẩn cho các chỉ số thống kê, cả trên toàn giải đấu và theo từng đội, sau đó xuất ra tệp CSV.
- **Chi tiết**:
  - **Tham số**: df, columns, output_file (mặc định: Bài 2/results2.csv).
  - **Tạo thư mục**: Tạo thư mục Bài 2 nếu chưa tồn tại.
  - **Xử lý dữ liệu**:
    - teams: Danh sách các đội duy nhất, thêm 'all (toàn bộ)' cho toàn giải đấu.
    - stats_data: Từ điển lưu kết quả thống kê.
    - Duyệt qua từng cột: Tính trung vị, trung bình, độ lệch chuẩn cho toàn giải và từng đội, bỏ qua cột không hợp lệ.
  - **Xuất kết quả**: Tạo DataFrame và lưu vào tệp CSV với mã hóa utf-8.
  - **Gọi hàm**: Áp dụng cho df và numeric_columns.
- **Ứng dụng**: Cung cấp thông tin thống kê tổng quan, hỗ trợ so sánh hiệu suất giữa các đội.

4. Hàm create_histograms

Mã Nguồn

```
def create_histograms(df, columns, teams):
    os.makedirs('Bài 2/histograms', exist_ok=True)
    for col in columns:
        if col not in df.columns or df[col].dropna().size < 3 or df[col].dropna().max() ==
df[col].dropna().min():
            continue
        plt.figure(figsize=(10, 6))
        n_bins = min(30, max(5, int(df[col].dropna().size / 5)))
        plt.hist(df[col].dropna(), bins=n_bins, density=True, histtype='bar', align='mid',
orientation='vertical', rwidth=0.8, color='blue', alpha=0.7)
        plt.title(f'Distribution of {col} - All Players')
        plt.xlabel(col)
        plt.ylabel('Density')
        plt.grid(True, alpha=0.3)
        safe_col = col.replace('/', '_').replace('%', 'pct').replace('\\', '_')
        plt.savefig(f'Bài 2/histograms/league_{safe_col}.png')
        plt.close()
    for team in teams:
        team_df = df[df['team'] == team]
        safe_team = ''.join(c if c.isalnum() else '_' for c in team)
        for col in columns:
```

```
        if col not in df.columns or team_df[col].dropna().size < 3 or
team_df[col].dropna().max() == team_df[col].dropna().min():
            continue
        plt.figure(figsize=(10, 6))
        n_bins = min(20, max(3, int(team_df[col].dropna().size / 3)))
        plt.hist(team_df[col].dropna(), bins=n_bins, density=True, histtype='bar', align='mid',
orientation='vertical', rwidth=0.8, color='green', alpha=0.7)
        plt.title(f'Distribution of {col} - {team}')
        plt.xlabel(col)
        plt.ylabel('Density')
        plt.grid(True, alpha=0.3)
        safe_col = col.replace('/', '_').replace('%', 'pct').replace('\\', '_')
        plt.savefig(f'Bài 2/histograms/{safe_team}_{safe_col}.png')
        plt.close()

teams = df['team'].unique()
create_histograms(df, numeric_columns, teams)
```

Giải Thích

- **Mục đích**: Tạo và lưu các biểu đồ histogram để hiển thị phân bố của các chỉ số thống kê trên toàn giải đấu và theo từng đội.
- **Chi tiết**:
  - **Tham số**: df, columns, teams (danh sách đội duy nhất từ df['team'].unique()).
  - **Tạo thư mục**: Tạo thư mục Bài 2/histograms nếu chưa tồn tại.
  - **Xử lý dữ liệu**:
    - Bỏ qua cột nếu số lượng giá trị hợp lệ < 3 hoặc không có sự thay đổi (max = min).
    - Tính số bin động dựa trên số lượng dữ liệu (tối đa 30 cho toàn giải, 20 cho từng đội).
    - Vẽ histogram cho toàn giải (màu xanh) và từng đội (màu xanh lá), chuẩn hóa mật độ, lưu vào tệp với tên an toàn (thay ký tự đặc biệt).
  - **Gọi hàm**: Áp dụng cho df, numeric_columns, và teams.
- **Ứng dụng**: Hỗ trợ hình dung phân bố dữ liệu, giúp phân tích xu hướng và độ lệch của từng chỉ số.

5. Hàm find_top_teams

Mã Nguồn

```
def find_top_teams(df, columns):
    team_stats = []
    for col in columns:
        if col not in df.columns or df[col].dropna().empty:
            continue
        team_means = df.groupby('team')[col].mean()
        if team_means.empty or team_means.isna().all():
            continue
        top_team = team_means.idxmax()
        top_value = team_means.max()
        if pd.isna(top_team) or pd.isna(top_value):
            continue
        team_stats.append({
            'Statistic': col,
```

```
            'Top Team': top_team,
            'Average Value': top_value
        })
    return pd.DataFrame(team_stats)

top_teams_df = find_top_teams(df, numeric_columns)
```
Giải Thích

- Mục đích: Xác định đội bóng có giá trị trung bình cao nhất cho mỗi chỉ số thống kê.
- Chi tiết:
  - Tham số: df, columns.
  - Xử lý dữ liệu:
    - Bỏ qua cột không hợp lệ.
    - Tính trung bình theo đội cho từng cột, lấy đội có giá trị cao nhất (idxmax()).
    - Lưu kết quả vào danh sách team_stats.
  - Trả về: DataFrame top_teams_df chứa đội dẫn đầu và giá trị trung bình.
  - Gọi hàm: Áp dụng cho df và numeric_columns.
- Ứng dụng: Hỗ trợ đánh giá hiệu suất đội bóng qua các chỉ số trung bình.

6. Hàm analyze_team_performance

Mã Nguồn

```
def analyze_team_performance(top_teams_df):
    os.makedirs('Bài 2', exist_ok=True)
    output_file = 'Bài 2/team_performance.txt'
    if top_teams_df.empty:
        with open(output_file, 'w', encoding='utf-8') as f:
            f.write("No team performance data available.")
        return
    team_counts = top_teams_df['Top Team'].value_counts()
    best_team = team_counts.index[0]
    best_team_count = team_counts.iloc[0]
    analysis = (
        f"Dựa trên phân tích, {best_team} dường như đang thi đấu tốt nhất trong mùa giải Ngoại hạng Anh 2024-2025.\n"
        f"Họ dẫn đầu ở {best_team_count} trên tổng số {len(top_teams_df)} hạng mục thống kê.\n"
        "Điều này cho thấy một màn trình diễn mạnh mẽ trên nhiều phương diện của trận đấu, bao gồm hiệu suất tấn công, "
        "khả năng triển khai bóng lên phía trước, và sự tham gia tổng thể vào các trận đấu."
    )
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write("Top Teams for Each Statistic:\n")
        for _, row in top_teams_df.iterrows():
            f.write(f"{row['Statistic']}: {row['Top Team']} (Avg: {row['Average Value']:.2f})\n")
        f.write("\nPerformance Analysis:\n")
        f.write(analysis)

analyze_team_performance(top_teams_df)
```
Giải Thích

- Mục đích: Phân tích và ghi nhận đội bóng dẫn đầu dựa trên số lần xuất hiện trong danh sách đội hàng đầu.
- Chi tiết:
  - Tham số: top_teams_df.
  - Tạo thư mục: Tạo thư mục Bài 2 nếu chưa tồn tại.
  - Xử lý dữ liệu:
    - Nếu top_teams_df rỗng, ghi thông báo và thoát.
    - Đếm số lần mỗi đội dẫn đầu, lấy đội xuất hiện nhiều nhất (best_team).
    - Tạo chuỗi phân tích dựa trên số lần dẫn đầu.
  - Ghi kết quả: Ghi danh sách đội dẫn đầu và phân tích vào tệp team_performance.txt.
  - Gọi hàm: Áp dụng cho top_teams_df.
- Ứng dụng: Đánh giá đội bóng xuất sắc nhất dựa trên hiệu suất tổng hợp qua các chỉ số thống kê.

III. Bài 3

1. Nhập Thư Viện và Tải Dữ Liệu

Mã Nguồn

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

plt.ioff()

df = pd.read_csv('Bài 1/result.csv')
numeric_columns = [
    'age', 'matches_played', 'starts', 'minutes', 'goals', 'assists',
    'yellow_cards', 'red_cards', 'xG', 'xAG', 'PrgC', 'PrgP', 'PrgR',
    'per90_Gls', 'per90_Ast', 'per90_xG', 'per90_xAG', 'SoT%', 'SoT/90',
    'G/Sh', 'Dist', 'Pass_Cmp', 'Pass_Cmp%', 'TotDist', 'Short_Cmp%',
    'Medium_Cmp%', 'Long_Cmp%', 'KP', '1/3', 'PPA', 'CrsPA', 'SCA', 'SCA90',
    'GCA', 'GCA90', 'Tkl', 'TklW', 'Challenges_Att', 'Challenges_Lost',
    'Blocks', 'Blocks_SH', 'Blocks_Pass', 'Blocks_Int', 'Touches', 'Def_Pen',
    'Def_3rd', 'Mid_3rd', 'Att_3rd', 'Att_Pen', 'Take_Att', 'Take_Succ%',
    'Take_Tkld%', 'Carries', 'Carries_ProDist', 'Carries_ProgC', 'Carries_1/3',
    'Carries_CPA', 'Carries_Mis', 'Carries_Dis', 'REC', 'REC_PrgR', 'Fls',
    'Fld', 'Off', 'Crs', 'OG', 'Recov', 'Aerial_Won', 'Aerial_Lost', 'Aerial_Won%'
]
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

Giải Thích

- Mục đích: Nhập các thư viện cần thiết, tải dữ liệu từ tệp CSV, và chuẩn hóa các cột số để chuẩn bị cho phân tích và phân cụm.
- Chi tiết:
    - Nhập thư viện:
        - pandas (bí danh pd): Dùng để xử lý và phân tích dữ liệu dạng bảng (DataFrame).
        - numpy (bí danh np): Hỗ trợ tính toán số học và thống kê.
        - matplotlib.pyplot (bí danh plt): Dùng để tạo biểu đồ, như biểu đồ Elbow và histogram.
        - os: Hỗ trợ thao tác với hệ thống tệp, như tạo thư mục.
        - sklearn.cluster.KMeans: Thuật toán phân cụm K-Means để nhóm các cầu thủ.
        - sklearn.preprocessing.StandardScaler: Chuẩn hóa dữ liệu (đưa về giá trị trung bình 0 và độ lệch chuẩn 1).
        - sklearn.decomposition.PCA: Phân tích thành phần chính (PCA) để giảm chiều dữ liệu và trực quan hóa.
        - seaborn (bí danh sns): Tạo biểu đồ phân tán với màu sắc theo cụm.
    - Tắt chế độ hiển thị tương tác của Matplotlib:
        - plt.ioff(): Ngăn Matplotlib hiển thị biểu đồ trực tiếp, đảm bảo chỉ lưu biểu đồ vào tệp.
    - Tải dữ liệu:
        - pd.read_csv('Bài 1/result.csv'): Đọc tệp CSV chứa dữ liệu cầu thủ (được tạo từ chương trình trước) vào DataFrame df.
    - Định nghĩa cột số:
        - numeric_columns: Danh sách các cột chứa dữ liệu số (tuổi, số trận, bàn thắng, xG, v.v.).
    - Chuẩn hóa cột số:
        - Vòng lặp for sử dụng pd.to_numeric để chuyển đổi các cột trong numeric_columns thành kiểu số, với errors='coerce' để thay các giá trị không hợp lệ (như "N/a") bằng NaN.
- Ứng dụng: Chuẩn bị dữ liệu cho các bước phân cụm và phân tích, đảm bảo các cột số có định dạng phù hợp.

2. Chuẩn Bị Dữ Liệu và Chuẩn Hóa

3. Phương Pháp Elbow để Chọn Số Cụm Tối Ưu

Mã Nguồn

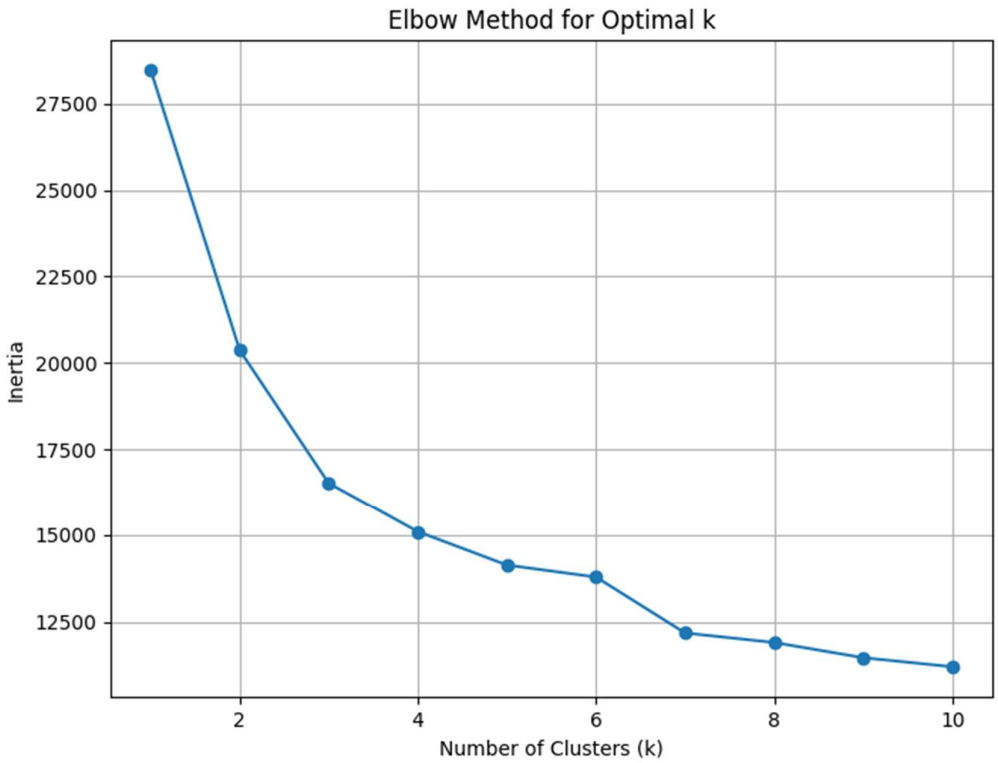
```
inertia = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 6))
plt.plot(k_range, inertia, marker='o')
```

```
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.grid(True)
plt.savefig('Bài 3/elbow_plot.png')
plt.close()
```

Giải Thích

- Mục đích: Sử dụng phương pháp Elbow để xác định số cụm tối ưu (k) cho thuật toán K-Means và trực quan hóa kết quả.
- Chi tiết:
    - Tính inertia:
        - Khởi tạo danh sách rỗng inertia để lưu giá trị inertia (tổng bình phương khoảng cách từ mỗi điểm đến trung tâm cụm gần nhất).
        - k_range = range(1, 11): Thử các giá trị k từ 1 đến 10.
        - Vòng lặp for:
            - Khởi tạo KMeans với số cụm k và random_state=42 để đảm bảo kết quả có thể tái lập.
            - Huấn luyện mô hình trên scaled_data bằng kmeans.fit(scaled_data).
            - Thêm giá trị inertia_ vào danh sách inertia.
    - Vẽ biểu đồ Elbow:
        - Tạo biểu đồ với kích thước 8x6 inch.
        - Vẽ đường cong với k_range trên trục x và inertia trên trục y, với điểm đánh dấu hình tròn (marker='o').
        - Đặt tiêu đề, nhãn trục, và lưới.
        - Lưu biểu đồ vào Bài 3/elbow_plot.png và đóng biểu đồ (plt.close()).
- Ứng dụng: Biểu đồ Elbow giúp xác định số cụm tối ưu bằng cách tìm "điểm khuỷu tay" (elbow point), nơi inertia giảm chậm lại. Trong trường hợp này, người dùng có thể quan sát biểu đồ để chọn k.

Hình ảnh minh họa:



4. Phân Cụm với K-Means

Mã Nguồn

```
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(scaled_data)

df_clusters = df.loc[data_for_clustering.index].copy()
df_clusters['Cluster'] = clusters
```

Giải Thích

- Mục đích: Áp dụng thuật toán K-Means với số cụm tối ưu để nhóm các cầu thủ và gán nhãn cụm vào DataFrame.
- Chi tiết:
    - Chọn số cụm:
        - optimal_k = 4: Số cụm được chọn (có thể dựa trên biểu đồ Elbow hoặc quyết định trước).
    - Phân cụm:
        - Khởi tạo KMeans với optimal_k cụm và random_state=42.
        - kmeans.fit_predict(scaled_data): Huấn luyện mô hình trên scaled_data và trả về mảng clusters chứa nhãn cụm (0, 1, 2, 3) cho mỗi cầu thủ.
    - Gán nhãn cụm:

        - df_clusters = df.loc[data_for_clustering.index].copy(): Tạo bản sao của DataFrame gốc, chỉ lấy các hàng đã được sử dụng trong phân cụm (loại bỏ hàng có NaN).
        - df_clusters['Cluster'] = clusters: Thêm cột Cluster chứa nhãn cụm vào df_clusters.
- Ứng dụng: Phân cụm giúp nhóm các cầu thủ có đặc điểm tương đồng (dựa trên các chỉ số thống kê) thành các nhóm, hỗ trợ phân tích vai trò và hiệu suất.

5. Tóm Tắt Thống Kê Theo Cụm

Mã Nguồn

```
cluster_summary = df_clusters.groupby('Cluster')[numeric_columns].mean()
cluster_summary.to_csv('Bài 3/cluster_summary.csv', encoding='utf-8-sig')
```

Giải Thích

- Mục đích: Tính trung bình các chỉ số thống kê cho từng cụm và xuất kết quả ra tệp CSV.
- Chi tiết:
    - Tính trung bình theo cụm:
        - df_clusters.groupby('Cluster'): Nhóm dữ liệu theo cột Cluster.
        - [numeric_columns].mean(): Tính giá trị trung bình của các cột số cho mỗi cụm.
        - Kết quả là DataFrame cluster_summary với chỉ số là các cụm (0, 1, 2, 3) và cột là các chỉ số thống kê.
    - Xuất ra CSV:
        - cluster_summary.to_csv: Lưu DataFrame vào tệp Bài 3/cluster_summary.csv với mã hóa utf-8-sig (hỗ trợ hiển thị ký tự tiếng Việt trên các hệ thống như Windows).
- Ứng dụng: Tệp cluster_summary.csv cung cấp cái nhìn tổng quan về đặc điểm trung bình của từng cụm, giúp hiểu rõ mỗi cụm đại diện cho loại cầu thủ nào (ví dụ: cầu thủ tấn công, cầu thủ phòng ngự, v.v.).

6. Giảm Chiều Dữ Liệu và Trực Quan Hóa với PCA

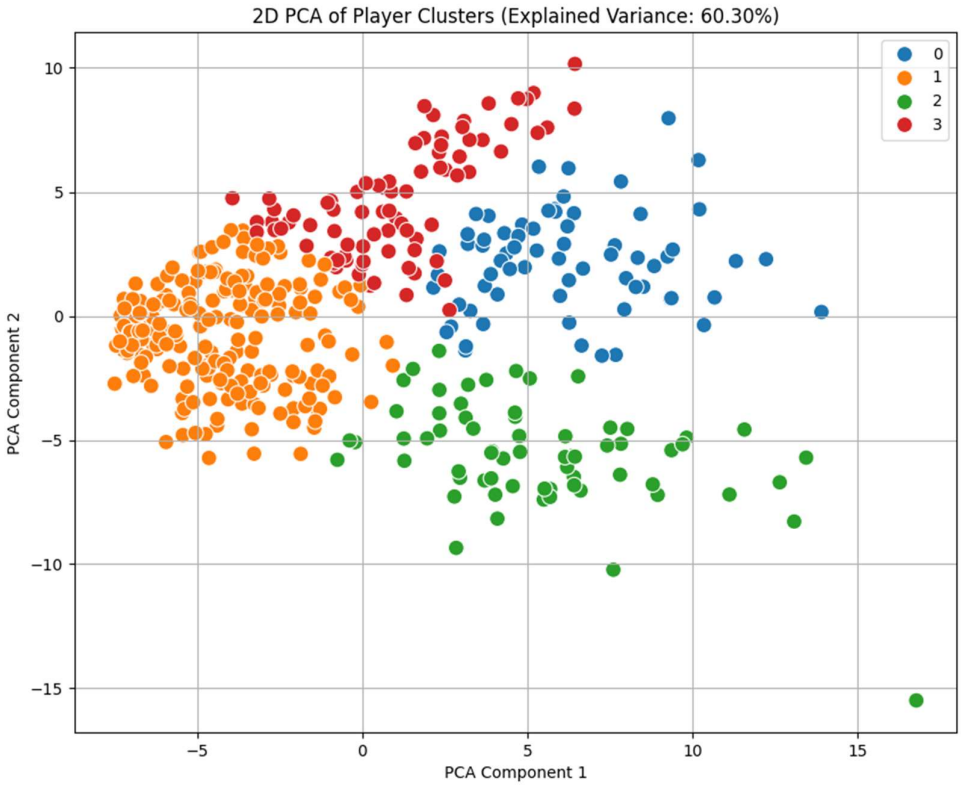
Mã Nguồn

```
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

explained_variance = pca.explained_variance_ratio_.sum() * 100

plt.figure(figsize=(10, 8))
sns.scatterplot(x=pca_data[:, 0], y=pca_data[:, 1], hue=clusters, palette='tab10', s=100)
plt.title(f'2D PCA of Player Clusters (Explained Variance: {explained_variance:.2f}%)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.grid(True)
plt.savefig('Bài 3/pca_clusters.png')
plt.close()
```

Giải Thích

- Mục đích: Giảm chiều dữ liệu xuống 2 chiều bằng PCA và trực quan hóa các cụm dưới dạng biểu đồ phân tán.
- Chi tiết:
    - Giảm chiều dữ liệu:
        - pca = PCA(n_components=2): Khởi tạo PCA để giảm chiều dữ liệu xuống 2 chiều.
        - pca_data = pca.fit_transform(scaled_data): Áp dụng PCA lên dữ liệu đã chuẩn hóa, trả về mảng pca_data với 2 cột (thành phần chính 1 và 2).
    - Tính tỷ lệ phương sai giải thích:
        - explained_variance = pca.explained_variance_ratio_.sum() * 100: Tính tổng tỷ lệ phương sai được giải thích bởi 2 thành phần chính (phần trăm thông tin được giữ lại).
    - Vẽ biểu đồ phân tán:
        - Tạo biểu đồ với kích thước 10x8 inch.
        - sns.scatterplot: Vẽ biểu đồ phân tán với:
            - Trục x: Thành phần chính 1 (pca_data[:, 0]).
            - Trục y: Thành phần chính 2 (pca_data[:, 1]).
            - Màu sắc theo cụm (hue=clusters), sử dụng bảng màu tab10.
            - Kích thước điểm 100 (s=100).
        - Đặt tiêu đề, nhãn trục, và lưới.
        - Lưu biểu đồ vào Bài 3/pca_clusters.png và đóng biểu đồ.
- Ứng dụng: Biểu đồ PCA giúp trực quan hóa các cụm trong không gian 2 chiều, cho thấy mức độ tách biệt giữa các cụm và phân bố của cầu thủ.

Hình ảnh minh họa:



7. Hàm find_extremes

Mã Nguồn

```
def find_extremes(df, columns, output_file='top_3.txt'):
    with open(output_file, 'w', encoding='utf-8-sig') as f:
        for col in columns:
            top_players = df[['name', 'team', col]].dropna().sort_values(by=col,
ascending=False).head(3)
            bottom_players = df[['name', 'team', col]].dropna().sort_values(by=col,
ascending=True).head(3)

            f.write(f"\nStatistic: {col}\n")
            f.write("Top 3 Players:\n")
            for _, row in top_players.iterrows():
                f.write(f"{row['name']} ({row['team']}): {row[col]:.2f}\n")
            f.write("Bottom 3 Players:\n")
            for _, row in bottom_players.iterrows():
                f.write(f"{row['name']} ({row['team']}): {row[col]:.2f}\n")
            f.write("-" * 50 + "\n")

find_extremes(df, numeric_columns)
```

Giải Thích

- Mục đích: Tìm và ghi ra tệp văn bản 3 cầu thủ có giá trị cao nhất và thấp nhất cho mỗi cột thống kê số.
- Chi tiết:
    - Tham số:
        - df: DataFrame chứa dữ liệu cầu thủ.
        - columns: Danh sách các cột số.
        - output_file: Đường dẫn tệp đầu ra (mặc định: top_3.txt).
    - Xử lý:
        - Mở tệp ở chế độ ghi với mã hóa utf-8-sig.
        - Duyệt qua từng cột trong columns:
            - top_players: Lấy 3 cầu thủ có giá trị cao nhất (sắp xếp giảm dần và lấy 3 hàng đầu).
            - bottom_players: Lấy 3 cầu thủ có giá trị thấp nhất (sắp xếp tăng dần).
        - Ghi kết quả vào tệp: Tiêu đề thống kê, danh sách 3 cầu thủ hàng đầu và 3 cầu thủ thấp nhất, với giá trị làm tròn đến 2 chữ số thập phân.
    - Gọi hàm: Áp dụng cho DataFrame df và numeric_columns.
- Ứng dụng: Cung cấp thông tin về các cầu thủ có hiệu suất nổi bật hoặc kém nhất trong từng chỉ số.

8. Hàm calculate_stats

Mã Nguồn

```
def calculate_stats(df, columns, output_file='results2.csv'):
    teams = ['all (toàn bộ)'] + list(df['team'].unique())
    stats_data = { 'Team': teams }

    for col in columns:
        stats_data[f'Trung vị của {col}'] = []
        stats_data[f'Trung bình của {col}'] = []
        stats_data[f'Độ lệch chuẩn của {col}'] = []

        median = df[col].median()
        mean = df[col].mean()
        std = df[col].std()
        stats_data[f'Trung vị của {col}'].append(median)
        stats_data[f'Trung bình của {col}'].append(mean)
        stats_data[f'Độ lệch chuẩn của {col}'].append(std)

        for team in teams[1:]:
            team_df = df[df['team'] == team]
            median = team_df[col].median()
            mean = team_df[col].mean()
            std = team_df[col].std()
            stats_data[f'Trung vị của {col}'].append(median)
            stats_data[f'Trung bình của {col}'].append(mean)
            stats_data[f'Độ lệch chuẩn của {col}'].append(std)
```

```
    stats_df = pd.DataFrame(stats_data)
    stats_df.to_csv(output_file, index=False, encoding='utf-8-sig')

calculate_stats(df, numeric_columns)
```

Giải Thích

- Mục đích: Tính toán trung vị, trung bình, và độ lệch chuẩn cho các chỉ số thống kê, cả trên toàn giải đấu và theo từng đội, sau đó xuất ra tệp CSV.
- Chi tiết:
  - Tham số:
    - df, columns, output_file (mặc định: results2.csv).
  - Xử lý:
    - Tạo danh sách teams bao gồm 'all (toàn bộ)' và các đội duy nhất.
    - Khởi tạo từ điển stats_data để lưu kết quả.
    - Duyệt qua từng cột: Tính trung vị, trung bình, độ lệch chuẩn cho toàn giải đấu và từng đội, sau đó lưu vào stats_data.
    - Tạo DataFrame stats_df và xuất ra tệp CSV.
  - Gọi hàm: Áp dụng cho df và numeric_columns.
- Ứng dụng: Cung cấp thông tin thống kê tổng quan, hữu ích để so sánh hiệu suất giữa các đội.

9. Hàm create_histograms

Mã Nguồn

```
def create_histograms(df, columns, teams):
    os.makedirs('Bài 3/histograms', exist_ok=True)

    for col in columns:
        plt.figure(figsize=(10, 6))
        plt.hist(df[col].dropna(), bins=30, density=True, histtype='bar', align='mid',
orientation='vertical', rwidth=0.8, color='blue', alpha=0.7)
        plt.title(f'Distribution of {col} - All Players')
        plt.xlabel(col)
        plt.ylabel('Density')
        plt.grid(True, alpha=0.3)
        plt.savefig(f'Bài 3/histograms/league_{col}.png')
        plt.close()

    for team in teams:
        team_df = df[df['team'] == team]
        for col in columns:
            plt.figure(figsize=(10, 6))
            plt.hist(team_df[col].dropna(), bins=30, density=True, histtype='bar', align='mid',
orientation='vertical', rwidth=0.8, color='green', alpha=0.7)
            plt.title(f'Distribution of {col} - {team}')
            plt.xlabel(col)
            plt.ylabel('Density')
```

```
        plt.grid(True, alpha=0.3)
        plt.savefig(f'Bài 3/histograms/{team}_{col}.png')
        plt.close()

teams = df['team'].unique()
create_histograms(df, numeric_columns, teams)
```

Giải Thích

- Mục đích: Tạo và lưu các biểu đồ histogram để hiển thị phân bố của các chỉ số thống kê trên toàn giải đấu và theo từng đội.
- Chi tiết:
  - Tham số:
    - df, columns, teams (danh sách đội duy nhất từ df['team'].unique()).
  - Xử lý:
    - Tạo thư mục Bài 3/histograms.
    - Vẽ histogram cho toàn giải đấu (màu xanh) và từng đội (màu xanh lá), với 30 khoảng, mật độ chuẩn hóa, và lưu vào các tệp riêng.
  - Gọi hàm: Áp dụng cho df, numeric_columns, và teams.
- Ứng dụng: Histogram giúp hình dung phân bố dữ liệu, hỗ trợ phân tích xu hướng và độ lệch.

10. Hàm find_top_teams

Mã Nguồn

```
def find_top_teams(df, columns):
    team_stats = []
    for col in columns:
        team_means = df.groupby('team')[col].mean().sort_values(ascending=False)
        top_team = team_means.index[0]
        top_value = team_means.iloc[0]
        team_stats.append({
            'Statistic': col,
            'Top Team': top_team,
            'Average Value': top_value
        })

    team_stats_df = pd.DataFrame(team_stats)
    return team_stats_df

top_teams_df = find_top_teams(df, numeric_columns)
```

Giải Thích

- Mục đích: Xác định đội có giá trị trung bình cao nhất cho mỗi chỉ số thống kê.
- Chi tiết:
  - Tham số:
    - df, columns.
  - Xử lý:
    - Duyệt qua từng cột: Nhóm theo đội, tính trung bình, sắp xếp giảm dần, lấy đội có giá trị cao nhất.

        - Tạo DataFrame team_stats_df chứa kết quả.
    - Gọi hàm: Áp dụng cho df và numeric_columns, lưu vào top_teams_df.
- Ứng dụng: Hỗ trợ đánh giá hiệu suất đội bóng qua các chỉ số trung bình.

11. Hàm analyze_team_performance

Mã Nguồn

```
def analyze_team_performance(top_teams_df):
    team_counts = top_teams_df['Top Team'].value_counts()
    best_team = team_counts.index[0]
    best_team_count = team_counts.iloc[0]

    analysis = (
        f"Based on the analysis, {best_team} appears to be performing the best in the 2024-2025 Premier League season.\n"
        f"They lead in {best_team_count} out of {len(numeric_columns)} statistical categories.\n"
        "This suggests strong performance across multiple aspects of the game, including offensive output, "
        "progressive play, and overall involvement in matches."
    )

    with open('team_performance.txt', 'w', encoding='utf-8-sig') as f:
        f.write("Top Teams for Each Statistic:\n")
        for _, row in top_teams_df.iterrows():
            f.write(f"{row['Statistic']}: {row['Top Team']} (Avg: {row['Average Value']:.2f})\n")
        f.write("\nPerformance Analysis:\n")
        f.write(analysis)

analyze_team_performance(top_teams_df)
```

Giải Thích

- Mục đích: Phân tích và ghi nhận đội bóng dẫn đầu dựa trên số lần xuất hiện trong danh sách đội hàng đầu.
- Chi tiết:
    - Tham số:
        - top_teams_df: DataFrame chứa đội dẫn đầu cho mỗi chỉ số.
    - Xử lý:
        - Đếm số lần mỗi đội xuất hiện, lấy đội dẫn đầu và số lần dẫn đầu.
        - Tạo chuỗi phân tích mô tả đội xuất sắc nhất.
        - Ghi danh sách đội dẫn đầu và phân tích vào tệp team_performance.txt.
    - Gọi hàm: Áp dụng cho top_teams_df.
- Ứng dụng: Đánh giá đội*i bóng xuất sắc nhất dựa trên các chỉ số thống kê.*

IV. Bài 4

1. Nhập Thư Viện và Chuẩn Bị Dữ Liệu

Mã Nguồn

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import undetected_chromedriver as uc
import time
import pandas as pd

csv_file = 'Bài 1/result.csv'
stats_df = pd.read_csv(csv_file)
filtered_players = stats_df[stats_df['minutes'] > 900][['name', 'team']].drop_duplicates()
filtered_players_set = set(zip(filtered_players['name'].str.strip(),
filtered_players['team'].str.strip()))
```

Giải Thích

- Mục đích: Nhập các thư viện cần thiết và chuẩn bị dữ liệu cầu thủ từ tệp CSV để lọc các cầu thủ có số phút thi đấu trên 900 phút.
- Chi tiết:
    - Nhập thư viện:
        - selenium.webdriver: Cung cấp công cụ để điều khiển trình duyệt.
        - selenium.webdriver.chrome.service.Service: Hỗ trợ khởi tạo dịch vụ Chrome.
        - selenium.webdriver.common.by.By: Cho phép định vị phần tử HTML bằng các phương thức như XPath hoặc tag name.
        - selenium.webdriver.support.ui.WebDriverWait: Hỗ trợ chờ cho đến khi phần tử xuất hiện.
        - selenium.webdriver.support.expected_conditions (bí danh EC): Cung cấp các điều kiện chờ.
        - undetected_chromedriver as uc: Sử dụng ChromeDriver không bị phát hiện để tránh bị chặn bởi các trang web.
        - time: Dùng để thêm độ trễ khi điều hướng.
        - pandas as pd: Dùng để đọc và xử lý dữ liệu từ tệp CSV.
    - Chuẩn bị dữ liệu:
        - csv_file = 'Bài 1/result.csv': Định nghĩa đường dẫn tệp CSV chứa dữ liệu cầu thủ.
        - stats_df = pd.read_csv(csv_file): Đọc tệp CSV vào DataFrame.
        - filtered_players = stats_df[stats_df['minutes'] > 900][['name', 'team']].drop_duplicates(): Lọc các cầu thủ có số phút thi đấu lớn hơn 900 phút, chọn cột name và team, và loại bỏ các bản sao.

        - filtered_players_set = set(zip(filtered_players['name'].str.strip(), filtered_players['team'].str.strip())): Chuyển danh sách cầu thủ thành tập hợp tuple (tên, đội) để kiểm tra nhanh.
- Ứng dụng: Chuẩn bị danh sách cầu thủ được lọc để tập trung vào những người có thời gian thi đấu đáng kể, đảm bảo dữ liệu thu thập có ý nghĩa.

2. Khởi Tạo Trình Duyệt và Điều Hướng

Mã Nguồn
```
driver = uc.Chrome()
wait = WebDriverWait(driver, 10)

link = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league'
driver.get(link)

time.sleep(3)

all_team_data = []

driver.execute_script("window.scrollBy(0, 300);")
time.sleep(1)
```
Giải Thích
- Mục đích: Khởi tạo trình duyệt không bị phát hiện, điều hướng đến trang web Football Transfers, và chuẩn bị để thu thập dữ liệu đội bóng.
- Chi tiết:
    - driver = uc.Chrome(): Khởi tạo trình duyệt Chrome không bị phát hiện.
    - wait = WebDriverWait(driver, 10): Tạo đối tượng chờ với thời gian tối đa 10 giây để đảm bảo phần tử tải xong.
    - link = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league': Định nghĩa URL của trang Premier League.
    - driver.get(link): Mở trang web.
    - time.sleep(3): Thêm độ trễ 3 giây để trang tải hoàn toàn.
    - all_team_data = []: Khởi tạo danh sách để lưu dữ liệu của tất cả các đội.
    - driver.execute_script("window.scrollBy(0, 300);"): Cuộn trang xuống 300 pixel để tải thêm nội dung (nếu có).
    - time.sleep(1): Thêm độ trễ 1 giây để đảm bảo nội dung cuộn được tải.
- Ứng dụng: Đảm bảo trình duyệt được khởi tạo và trang web được tải đúng cách, sẵn sàng cho việc trích xuất dữ liệu.

3. Xử Lý Cookie và Đợi Bảng Dữ Liệu

Mã Nguồn
```
try:
    cookie_buttons = driver.find_elements(By.XPATH, "//button[contains(text(), 'Accept') or contains(text(), 'Cookie') or contains(text(), 'Close')]")
    for button in cookie_buttons:
        driver.execute_script("arguments[0].click();", button)
        time.sleep(0.5)
```

```
except:
    pass

wait.until(EC.presence_of_element_located((By.TAG_NAME, 'tbody')))
teams_table = driver.find_element(By.TAG_NAME, 'tbody')
all_teams = teams_table.find_elements(By.TAG_NAME, 'tr')
```

Giải Thích

- Mục đích: Xử lý các thông báo cookie (nếu có) và chờ cho đến khi bảng dữ liệu đội bóng tải xong.
- Chi tiết:
  - Xử lý cookie:
    - cookie_buttons = driver.find_elements(By.XPATH, "..."): Tìm các nút liên quan đến cookie (chứa "Accept", "Cookie", hoặc "Close").
    - Vòng lặp for: Nhấp vào từng nút bằng JavaScript để chấp nhận cookie, với độ trễ 0.5 giây giữa các lần nhấp.
    - Khối try-except: Bỏ qua lỗi nếu không tìm thấy nút cookie.
  - Chờ bảng dữ liệu:
    - wait.until(...): Chờ cho đến khi phần tử <tbody> (bảng dữ liệu) xuất hiện.
    - teams_table = driver.find_element(By.TAG_NAME, 'tbody'): Lấy phần tử bảng.
    - all_teams = teams_table.find_elements(By.TAG_NAME, 'tr'): Lấy tất cả các hàng (đội bóng) trong bảng.
- Ứng dụng: Đảm bảo các thông báo cookie được xử lý và dữ liệu đội bóng sẵn sàng để trích xuất, tránh lỗi do tải trang không đầy đủ.

4. Thu Thập Dữ Liệu Cầu Thủ Theo Từng Đội

Mã Nguồn

```
for i in range(len(all_teams)):
    teams_table = driver.find_element(By.TAG_NAME, 'tbody')
    all_teams = teams_table.find_elements(By.TAG_NAME, 'tr')
    team = all_teams[i]

    team_name = team.find_elements(By.TAG_NAME, 'td')[2].text
    print(f"Processing team: {team_name}")

    try:
        team_link = team.find_element(By.TAG_NAME, 'a')
        team_url = team_link.get_attribute('href')

        driver.get(team_url)
    except Exception as e:
        print(f"Could not find team link, trying JavaScript click: {str(e)}")
        driver.execute_script("arguments[0].click();", team)

    time.sleep(3)
```

```
    player_data = []
    try:
        wait.until(EC.presence_of_element_located((By.TAG_NAME, 'tbody')))

        players_table = driver.find_element(By.TAG_NAME, 'tbody')
        players = players_table.find_elements(By.TAG_NAME, 'tr')

        print(f"  Found {len(players)} players")

        for player in players:
            try:
                player_name_elem = player.find_element(By.XPATH, ".//th[@class='td-
player']//a")
                player_name = player_name_elem.text.strip()

                if (player_name, team_name) not in filtered_players_set:
                    print(f"  - Skipping player: {player_name} (not in filtered list or under 900
minutes)")
                    continue

                try:
                    transfer_value = player.find_element(By.CLASS_NAME, 'player-tag').text
                except:
                    try:
                        transfer_value = player.find_element(By.XPATH, ".//td[contains(@class,
'value')]").text
                    except:
                        transfer_value = "Not available"

                player_data.append({
                    'Team': team_name,
                    'Player': player_name,
                    'Transfer Value': transfer_value
                })
                print(f"  - Added player: {player_name}, Value: {transfer_value}")
            except Exception as e:
                print(f"  - Error extracting player data: {str(e)}")

    except Exception as e:
        print(f"Error finding players: {str(e)}")

    all_team_data.extend(player_data)

    driver.get(link)
    time.sleep(3)
```

```
    driver.execute_script("window.scrollBy(0, 300);")
    time.sleep(1)
```
Giải Thích

- Mục đích: Thu thập dữ liệu giá trị chuyển nhượng của các cầu thủ từ trang chi tiết của từng đội.
- Chi tiết:
    - Duyệt qua các đội:
        - Vòng lặp for qua các hàng all_teams, lấy đội tại chỉ số i.
        - team_name = team.find_elements(By.TAG_NAME, 'td')[2].text: Lấy tên đội từ cột thứ 3.
        - In thông báo xử lý đội hiện tại.
    - Điều hướng đến trang đội:
        - Thử lấy liên kết đội bằng team.find_element(By.TAG_NAME, 'a') và điều hướng đến team_url.
        - Nếu thất bại, sử dụng JavaScript để nhấp vào đội (driver.execute_script).
        - Thêm độ trễ 3 giây để trang tải.
    - Thu thập dữ liệu cầu thủ:
        - Chờ bảng cầu thủ (<tbody>).
        - Lấy danh sách cầu thủ từ bảng.
        - Duyệt qua từng cầu thủ:
            - Lấy tên cầu thủ từ th[@class='td-player']//a.
            - Kiểm tra nếu cầu thủ thuộc filtered_players_set, nếu không thì bỏ qua.
            - Thử lấy giá trị chuyển nhượng từ player-tag, nếu không thành công thì từ td[contains(@class, 'value')], nếu vẫn thất bại thì gán "Not available".
            - Thêm dữ liệu (đội, tên, giá trị) vào player_data.
        - Kết hợp player_data vào all_team_data.
    - Quay lại trang chính:
        - Quay lại link, cuộn trang, và thêm độ trễ.
- Ứng dụng: Thu thập dữ liệu giá trị chuyển nhượng từ các trang chi tiết, chỉ tập trung vào cầu thủ đã lọc, đảm bảo dữ liệu chính xác và đầy đủ.

5. Lưu Dữ Liệu và Đóng Trình Duyệt

Mã Nguồn
```
df = pd.DataFrame(all_team_data)

df.to_csv('Bài 4/premier_league_player_values.csv', index=False)
print(f"Data saved to 'premier_league_player_values.csv'. Total players:
{len(all_team_data)}")

driver.quit()
```
Giải Thích

- Mục đích: Chuyển đổi dữ liệu thành DataFrame và lưu vào tệp CSV, sau đó đóng trình duyệt.
- Chi tiết:
    - df = pd.DataFrame(all_team_data): Tạo DataFrame từ danh sách all_team_data.
    - df.to_csv('Bài 4/premier_league_player_values.csv', index=False): Lưu DataFrame vào tệp CSV, không bao gồm chỉ số hàng.
    - print(...): In thông báo xác nhận, hiển thị số lượng cầu thủ được thu thập.
    - driver.quit(): Đóng trình duyệt để giải phóng tài nguyên.
- Ứng dụng: Đảm bảo dữ liệu được lưu trữ an toàn và trình duyệt được đóng đúng cách sau khi hoàn thành.