HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

✧✧✧🕮✧✧✧

BÁO CÁO BÀI TẬP LỚN

Môn học: Lập trình với Python

Giảng viên: Kim Ngọc Bách

Sinh viên: Nguyễn Bá Dương

Mã sinh viên: B22DCCN165

Ngày sinh: 08/02/2004

Lớp: D22CQCN09-B

Số điện thoại: 0815750059

Hà Nội, 2024

MỤC LỤC

Câu 1: Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.

- **Nguồn dữ liệu: https://fbref.com/en/**
- **Ghi kết quả ra file 'results.csv', bảng kết quả có cấu trúc như sau:**
  - **Mỗi cột tương ứng với một chỉ số.**
  - **Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.**
  - **Chỉ số nào không có hoặc không áp dụng thì để là N/a**
- **[*] Các chỉ số cần thu thập như sau:**
  - **Nation**
  - **Team**
  - **Position**
  - **Age**
  - **Playing time: matches played, starts, minutes**
  - **Performance: non-Penalty Goals, Penalty Goals, Assists, Yellow Cards, Red Cards.**
  - **Expected: xG, npxG, xAG**
  - **Progression: PrgC, PrgP, PrgR**
  - **Per 90 minutes: Gls, Ast, G+A, G-PK, G+A-PK, xG, xAG, xG + xAG, npxG, npxG + xAG**
  - **Goalkeeping:**
    - **Performance: GA, GA90, SoTA, Saves, Save%, W, D, L, CS, CS%**
    - **Penalty Kicks: PKatt, PKA, PKsv, PKm, Save%**
  - **Shooting:**
    - **Standard: Gls, Sh, SoT, SoT%, Sh/90, SoT/90, G/Sh, G/SoT, Dist, FK, PK, Pkatt**
    - **Expected: xG, npxG, npxG/Sh, G-xG, np:G-xG**
  - **Passing:**
    - **Total: Cmp, Att, Cmp%, TotDist, PrgDist**
    - **Short: Cmp, Att, Cmp%**
    - **Medium: Cmp, Att, Cmp%**
    - **Long: Cmp, Att, Cmp%**
    - **Expected: Ast, xAG, xA, A-xAG, KP, 1/3, PPA, CrsPA, PrgP**

+ **Pass Types:**
  - **Pass Types: Live, Dead, FK, TB, Sw, Crs, TI, CK**
  - **Corner Kicks: In, Out, Str**
  - **Outcomes: Cmp, Off, Blocks**
+ **Goal and Shot Creation**
  - **SCA: SCA, SCA90**
  - **SCA Types: PassLive, PassDead, TO, Sh, Fld, Def**
  - **GCA: GCA, GCA90**
  - **GCA Types: PassLive, PassDead, TO, Sh, Fld, Def**
+ **Defensive Actions:**
  - **Tackles: Tkl, TklW, Def 3rd, Mid 3rd, Att 3rd**
  - **Challenges: Tkl, Att, Tkl%, Lost**
  - **Blocks: Blocks, Sh, Pass, Int, Tkl + Int, Clr, Err**
+ **Possession:**
  - **Touches: Touches, Def Pen, Def 3rd, Mid 3rd, Att 3rd, Att Pen, Live**
  - **Take-Ons: Att, Succ, Succ%, Tkld, Tkld%**
  - **Carries: Carries, TotDist, ProDist, ProgC, 1/3, CPA, Mis, Dis**
  - **Receiving: Rec, PrgR**
+ **Playing Time:**
  - **Starts: Starts, Mn/Start, Compl**
  - **Subs: Subs, Mn/Sub, unSub**
  - **Team Success: PPM, onG, onGA**
  - **Team Success xG: onxG, onxGA**
+ **Miscellaneous Stats:**
  - **Performance: Fls, Fld, Off, Crs, OG, Recov**
  - **Aerial Duels: Won, Lost, Won%**
+ **Tham khảo: https://fbref.com/en/squads/822bd0ba/2023-2024//en/squads/822bd0ba/2023-2024/Liverpool-Stats**

1. Giới thiệu

– Trong bóng đá hiện đại, việc phân tích dữ liệu cầu thủ dựa trên các chỉ số thống kê là một công cụ quan trọng để giúp các đội bóng có thể đưa ra các chiến thuật và cách thức xây dựng đội hình. Bài tập này nhằm mục đích thu

thập dữ liệu cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá Ngoại Hạng Anh mùa giải 2023-2024 từ nguồn dữ liệu https://fbref.com/en/

- Để thực hiện bài tập này, em đã sử dụng Selenium. Selenium là một công cụ tự động hoá trình duyệt web phổ biến được sử dụng cho việc kiểm thử tự động trên ứng dụng web và tự động hoá các tác vụ trên trình duyệt web. Selenium cho phép người dùng thực hiện các hành động như điều khiển trình duyệt web, nhập liệu… Và trong bài này em đã sử dụng công cụ Selenium WebDriver.
- Bên cạnh đó, em cũng sử dụng thư viện Beautifulsoup để phân tích cú pháp HTML, giúp việc trích xuất thông tin từ trang web trở nên dễ dàng hơn.
- Cuối cùng, em sử dụng thư viện Pandas để dễ dàng làm việc với dữ liệu có cấu trúc dạng bảng như bảng tính Excel. Em lựa chọn sử dụng cấu trúc dữ liệu của Pandas là DataFrame. Đó là cấu trúc dữ liệu hai chiều kiểu dạng bảng dữ liệu, nó cho phép áp dụng các thuật toán trên dòng và cột.

2. Giải thích bài làm

2.1. Các thư viện và công cụ cần thiết

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
from time import sleep
```

- Đầu tiên, em import thư viện **pandas** để xử lý và phân tích dữ liệu. Nó cung cấp các công cụ hữu ích để ta đọc và ghi file csv.
- **From selenium import webdriver:**
  - **Selenium:** Đây là một công cụ tự động hoá trình duyệt web, nó được dùng để thu thập thông tin web.
  - **Webdriver:** Đây là một module của selenium giúp khởi tạo trình duyệt và thực hiện các thao tác tự động.
- **From selenium.webdriver.chrome.service import Service:** Service là một lớptrong selenium giúp khởi tạo trình điều khiển cho trình duyệt Chrome. Lí do em dùng trình duyệt Chrome là bởi vì máy em cài đặt trình duyệt Chrome.
- **From webdriver_manager.chrome import ChromeDriverManager:** Webdriver_manager là một thư viện hỗ trợ tải tự động ChromeDriver còn

ChromeDriverManager giúp chúng ta có thể xác định được phiên bản ChromeDriver phù hợp với phiên bản trình duyệt Chrome.

- **From bs4 import BeautifulSoup:** Trong bài này em sử dụng thư viện BeautifulSoup để phân tích cú pháp HTML, lọc và tìm kiếm nội dung trong mã HTML mà selenium đã lấy được.
- **From time import sleep**: Em sử dụng hàm sleep trong thư viện time để cho chương trình được phép dừng trong một khoảng thời gian nhất định để tránh việc trang web chặn truy cập tự động.

2.2. Cấu hình và khởi tạo Selenium

```
# Cấu hình Selenium để sử dụng Chrome
options_chrome = webdriver.ChromeOptions()
# tự động quản lý phiên bản driver
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=options_chrome)
```

- options_chrome: Chứa các tuỳ chọn cấu hình Chrome.
- Tự động quản lý phiên bản driver bằng cách tự động tải ChromeDriver và đảm bảo rằng phiên bản ChromeDriver phù hợp với phiên bản trình duyệt Chrome trên máy thực hiện bài tập.
- Cuối cùng là khởi tạo đối tượng driver đại diện cho trình duyệt Chrome để giúp cho Selenium có thể điều khiển các thao tác trên trang web.

2.3. Thu thập dữ liệu cầu thủ

- Xác định các thuộc tính cần lấy từ bảng: Với yêu cầu của đề bài cần thu thập các chỉ số như trên, em sử dụng một danh sách có tên là **players** để lưu các chỉ số đó của cầu thủ như tên, tuổi, quốc tịch,…

```
players = [
    'Name', 'Nation','Team', 'Position', 'Age',
    'Matches', 'Starts', 'Minutes',
    'non_penalty_goals', 'penalty_goals', 'assists', 'yellow_cards',
    'red_cards',
    'xG', 'npxG', 'xAG',
    'PrgC', 'PrgP', 'PrgR',
    'per90_Gls', 'per90_Ast', 'per90_G+A', 'per90_G-PK',
    'per90_G+A-PK', 'per90_xG', 'per90_xAG', 'per90_xG+xAG',
    'per90_npxG', 'per90_npxG+xAG',
    'GA', 'GA90', 'SoTA', 'Saves',
    'Save%', 'W', 'D', 'L', 'CS', 'CS%',
    'PKatt', 'PKA', 'PKsv', 'PKm', 'GK_Save%',
    'Gls', 'Sh', 'SoT', 'SoT%', 'Sh/90', 'SoT/90', 'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt',
    'xG_shooting', 'npxG_shooting', 'npxG/Sh', 'G-xG', 'np:G-xG',
    'Pass_Cmp', 'Pass_Att', 'Pass_Cmp%', 'TotDist', 'PrgDist',
    'Short_Cmp', 'Short_Att', 'Short_Cmp%', 'Medium_Cmp',
    'Medium_Att', 'Medium_Cmp%', 'Long_Cmp', 'Long_Att',
    'Long_Cmp%', 'Ast', 'xAG', 'xA', 'A-xAG', 'KP', '1/3', 'PPA',
    'CrsPA', 'PrgP', 'Pass_Live', 'Pass_Dead', 'Pass_FK',
    'Pass_TB', 'Pass_Sw', 'Pass_Crs', 'Pass_TI', 'Pass_CK',
    'Corner_In','Corner_Out','Corner_Str',
    'Pass_Cmp_outcome', 'Pass_Off', 'Pass_Blocks',
    'SCA','SCA90','SCA_type_Passlive','SCA_type_Passdead','SCA_type_TO','SCA_type_Sh','SCA_type_Fld','SCA_type_Def',
    'GCA','GCA90','GCA_type_Passlive','GCA_type_Passdead','GCA_type_TO','GCA_type_Sh','GCA_type_Fld','GCA_type_Def',
    'Tkl','TklW', 'Def_3rd', 'Mid_3rd', 'Att_3rd',
    'Challenges_Tkl','Challenges_Att', 'Challenges_Tkl%', 'Challenges_Lost',
    'Blocks','Blocks_Sh','Blocks_Pass','Blocks_Int','Blocks_Tkl+Int','Blocks_Clr','Blocks_Err',
    'Touches', 'Def_Pen', 'Def_3rd', 'Mid_3rd', 'Att_3rd',
```

– Phân tích trang web:

- Ta bắt đầu mở trang web đầu tiên lên và chờ trang tải lên:

```
# Mở trang web 1
driver.get("https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats")
sleep(1)
```

- Lấy HTML và sử dụng BeautifulSoup để phân tích

```
# Lấy HTML sau khi trang web đã được tải lên
html = driver.page_source
# phân tích cú pháp HTML
soup = BeautifulSoup(html, 'html.parser')
```

- Truy xuất bảng dữ liệu cầu thủ:

Ở trên trang web, em đã tìm được bảng dữ liệu của cầu thủ nằm trong thẻ talbe có id là ‘stats_standard’, chính vì thế em tìm kiếm bảng có id là ‘stats_standard’ và sử dụng danh sách có tên là data để lưu dữ liệu cầu thủ.

```
table = soup.find('table', {'id': 'stats_standard'})
# Khởi tạo danh sách để lưu dữ liệu cầu thủ
data = []
```

- Trích xuất dữ liệu:

Ban đầu, em khởi tạo cho tất cả các thuộc tính giá trị mặc định là ‘N/a’ để xử lý trường hợp có những thuộc tính không xuất hiện trong trang web này.

Tiếp đến, em tìm tất cả các thẻ ‘tr’ trong phần thân của thẻ ‘tbody’ của bảng dữ liệu cầu thủ và em lặp qua từng hàng để trích xuất dữ liệu của cầu thủ. Trong mỗi thẻ ‘tr’ đó em tìm tất cả các thẻ ‘td’ chính là nơi chứa thông tin về các chỉ số của cầu thủ.

Sau đó tiến hành lấy thông tin chi tiết từ mỗi hàng của bảng.

```
# Tìm tất cả các hàng trong tbody (bỏ qua hàng tiêu đề trong thead)
rows = table.tbody.find_all('tr')
# Duyệt qua từng hàng và lấy dữ liệu
for row in rows:
    cols = row.find_all('td')
    if not cols:
        continue
    # Ban đầu ta cho các thuộc tính mặc định có giá trị "N/a"
    player = {col: "N/a" for col in players}

    # Bắt đầu lấy dữ liệu
    player['Name'] = cols[0].text.strip() if cols[0].text.strip() else "N/a"
    player['Nation'] = cols[1].text.strip() if cols[1].text.strip() else "N/a"
    player['Position'] = cols[2].text.strip() if cols[2].text.strip() else "N/a"
    player['Team'] = cols[3].text.strip() if cols[3].text.strip() else "N/a"
    player['Age'] = cols[4].text.strip() if cols[4].text.strip() else "N/a"
    player['Matches'] = cols[6].text.strip() if cols[6].text.strip() else "N/a"
    player['Starts'] = cols[7].text.strip() if cols[7].text.strip() else "N/a"
    player['Minutes'] = cols[8].text.strip() if cols[8].text.strip() else "N/a"
    player['assists'] = cols[11].text.strip() if cols[11].text.strip() else "N/a"
    player['non_penalty_goals'] = cols[13].text.strip() if cols[13].text.strip() else "N/a"
    player['penalty_goals'] = cols[14].text.strip() if cols[14].text.strip() else "N/a"
    player['yellow_cards'] = cols[16].text.strip() if cols[16].text.strip() else "N/a"
    player['red_cards'] = cols[17].text.strip() if cols[17].text.strip() else "N/a"
    player['xG'] = cols[18].text.strip() if cols[18].text.strip() else "N/a"
    player['npxG'] = cols[19].text.strip() if cols[19].text.strip() else "N/a"
    player['xAG'] = cols[20].text.strip() if cols[20].text.strip() else "N/a"
    player['PrgC'] = cols[22].text.strip() if cols[22].text.strip() else "N/a"
    player['PrgP'] = cols[23].text.strip() if cols[23].text.strip() else "N/a"
    player['PrgR'] = cols[24].text.strip() if cols[24].text.strip() else "N/a"
    player['per90_Gls'] = cols[25].text.strip() if cols[25].text.strip() else "N/a"
    player['per90_Ast'] = cols[26].text.strip() if cols[26].text.strip() else "N/a"
    player['per90_G+A'] = cols[27].text.strip() if cols[27].text.strip() else "N/a"
    player['per90_G-PK'] = cols[28].text.strip() if cols[28].text.strip() else "N/a"
    player['per90_G+A-PK'] = cols[29].text.strip() if cols[29].text.strip() else "N/a"
```

– Kiểm tra điều kiện: Với yêu cầu của đề bài là chỉ lấy các cầu thủ có số phút thi đấu nhiều hơn 90 phút, ta sẽ tiến hành kiểm tra điều kiện và thêm vào danh sách **data**

```
    # bỏ dấu phẩy trong cột minutes
    minutes_value = player['Minutes'].replace(',', '')

    # Chỉ thêm cầu thủ có số phút thi đấu nhiều hơn 90
    if minutes_value != "N/a" and int(minutes_value) > 90:
        data.append(player)
```

– Đối với các trang web còn lại:
  - Do đã lập danh sách các thuộc tính cần có của cầu thủ nên từ trang web thứ 2 (Goalkeeper) trở đi ta chỉ cần cập nhật chỉ số cho các thuộc tính có trong trang web đó.
  - Em xây dựng 1 danh sách mới để lưu chỉ số của các thuộc tính cần cập nhật ở trong trang web. Ví dụ đối với trang web cập nhật các chỉ số của Goalkeeper, em lập một danh sách có tên là **new_GK_data** để lưu các giá trị của các thuộc tính có trong trang web này. Sau đó tiến hành trích xuất dữ liệu như bình thường và lưu vào danh sách **new_GK_data** .

```
# Duyệt qua từng hàng và lấy dữ liệu
for row in rows:
    cols = row.find_all('td')
    if not cols:
        continue
    player = {}

    player['Name'] = cols[0].text.strip() if cols[0].text.strip() else "N/a"
    player['GA'] = cols[10].text.strip() if cols[10].text.strip() else "N/a"
    player['GA90'] = cols[11].text.strip() if cols[11].text.strip() else "N/a"
    player['SoTA'] = cols[12].text.strip() if cols[12].text.strip() else "N/a"
    player['Saves'] = cols[13].text.strip() if cols[13].text.strip() else "N/a"
    player['Save%'] = cols[14].text.strip() if cols[14].text.strip() else "N/a"
    player['W'] = cols[15].text.strip() if cols[15].text.strip() else "N/a"
    player['D'] = cols[16].text.strip() if cols[16].text.strip() else "N/a"
    player['L'] = cols[17].text.strip() if cols[17].text.strip() else "N/a"
    player['CS'] = cols[18].text.strip() if cols[18].text.strip() else "N/a"
    player['CS%'] = cols[19].text.strip() if cols[19].text.strip() else "N/a"
    player['PKatt'] = cols[20].text.strip() if cols[20].text.strip() else "N/a"
    player['PKA'] = cols[21].text.strip() if cols[21].text.strip() else "N/a"
    player['PKsv'] = cols[22].text.strip() if cols[22].text.strip() else "N/a"
    player['PKm'] = cols[23].text.strip() if cols[23].text.strip() else "N/a"
    player['GK_Save%'] = cols[24].text.strip() if cols[24].text.strip() else "N/a"

    new_GK_data.append(player)
```

  - Cuối cùng em cập nhật các chỉ số đó vào danh sách data ban đầu. Đối với các trang web còn lại em cũng làm tương tự.

```
# Danh sách các thuộc tính cần cập nhật
GK_attributes_to_update = ['GA', 'GA90', 'SoTA', 'Saves',
    'Save%', 'W', 'D', 'L', 'CS', 'CS%', 'PKatt', 'PKA', 'PKsv',
    'PKm', 'GK_Save%']

# Cập nhật vào danh sách "data" hiện có
for player in data:
    # Tìm cầu thủ trong new_data dựa vào tên
    matching_player = next((p for p in new_GK_data if p['Name'] == player['Name']), None)

    # Nếu tìm thấy cầu thủ, cập nhật các thuộc tính
    if matching_player:
        for attr in GK_attributes_to_update:
            player[attr] = matching_player[attr]
```

2.4. Lưu dữ liệu vào DataFrame và xuất ra file csv

– Lưu dữ liệu vào DataFrame

```
# Lưu dữ liệu vào DataFrame và xuất ra file CSV
df = pd.DataFrame(data)
```

– Sắp xếp thứ tự các cầu thủ theo yêu cầu:

- Đầu tiên em tách first name tạo thành một cột mới từ cột ‘Name’.

```
# Tạo cột 'First_Name' bằng cách tách tên đầy đủ
df['First_Name'] = df['Name'].apply(lambda x: x.split()[-1])
```

- Chuyển đổi cột ‘Age’ (tuổi) sang kiểu số nguyên (int) để dễ dàng sắp xếp

```
# Chuyển đổi cột 'Age' sang kiểu số nguyên (int) để có thể sắp xếp chính xác
df['Age'] = df['Age'].astype(int)
```

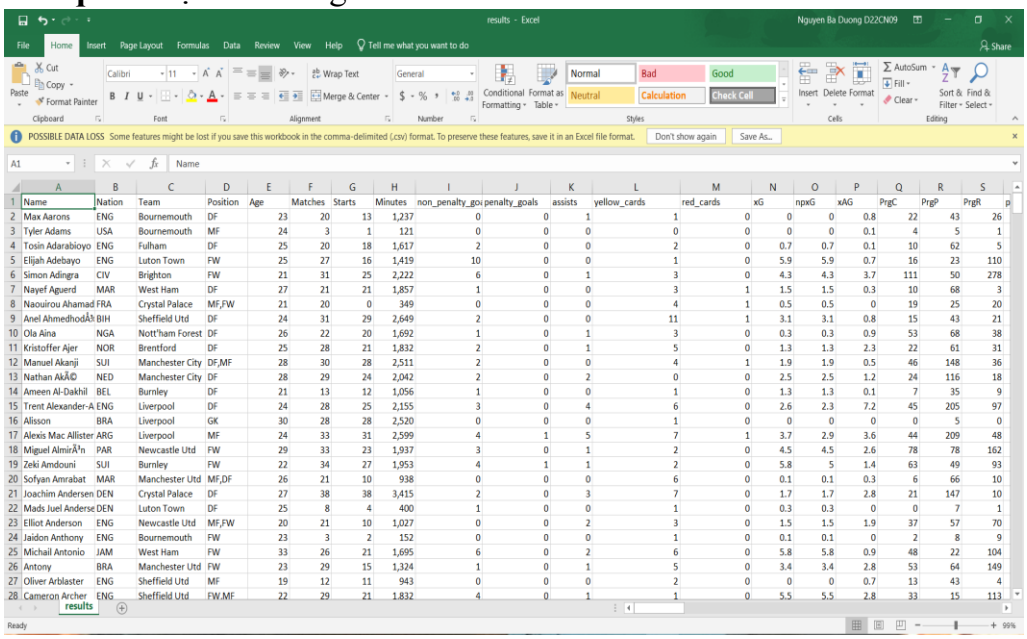
- Sắp xếp theo first name và sau đó theo tuổi từ lớn đến nhỏ

```
# Sắp xếp theo 'First_Name' và sau đó theo 'Age' từ lớn đến nhỏ
df = df.sort_values(by=['First_Name','Age'], ascending=[True, False])
```

– Cuối cùng xuất ra file ‘results.csv’ theo yêu cầu

```
df.to_csv('results.csv', index=False)
```

– **Kết quả**: được lưu trong file ‘results.csv’ :



Name	Nation	Team	Position	Age	Matches	Starts	Minutes	non_penalty_goals	penalty_goals	assists	yellow_cards	red_cards	xG	npxG	xAG	PrgC	PrgP	PrgR
Max Aarons	ENG	Bournemouth	DF	23	20	13	1,237	0	0	1	1	0	0	0	0.8	22	43	26
Tyler Adams	USA	Bournemouth	MF	24	3	1	121	0	0	0	0	0	0	0	0.1	4	5	1
Tosin Adarabioyo	ENG	Fulham	DF	25	20	18	1,617	2	0	0	2	0	0.7	0.7	0.1	10	62	5
Elijah Adebayo	ENG	Luton Town	FW	25	27	16	1,419	10	0	0	1	0	5.9	5.9	0.7	16	23	110
Simon Adingra	CIV	Brighton	FW	21	31	25	2,222	6	0	1	3	0	4.3	4.3	3.7	111	50	278
Nayef Aguerd	MAR	West Ham	DF	27	21	21	1,857	1	0	0	3	1	1.5	1.5	0.3	10	68	3
Naouirou Ahamad	FRA	Crystal Palace	MF,FW	21	20	0	349	0	0	0	4	1	0.5	0.5	0	19	25	20
Anel AhmedhodÅ¾	BIH	Sheffield Utd	DF	24	31	29	2,649	2	0	0	11	1	3.1	3.1	0.8	15	43	21
Ola Aina	NGA	Nott'ham Forest	DF	26	22	20	1,692	1	0	1	3	0	0.3	0.3	0.9	53	68	38
Kristoffer Ajer	NOR	Brentford	DF	25	28	21	1,832	2	0	1	5	0	1.3	1.3	2.3	22	61	31
Manuel Akanji	SUI	Manchester City	DF,MF	28	30	28	2,511	2	0	0	4	1	1.9	1.9	0.5	46	148	36
Nathan AkÃ©	NED	Manchester City	DF	28	29	24	2,042	2	0	2	0	0	2.5	2.5	1.2	24	116	18
Ameen Al-Dakhil	BEL	Burnley	DF	21	13	12	1,056	1	0	0	1	0	1.3	1.3	0.1	7	35	9
Trent Alexander-A	ENG	Liverpool	DF	24	28	25	2,155	3	0	4	6	0	2.6	2.3	7.2	45	205	97
Alisson	BRA	Liverpool	GK	30	28	28	2,520	0	0	0	1	0	0	0	0	0	5	0
Alexis Mac Allister	ARG	Liverpool	MF	24	33	31	2,599	4	1	5	7	1	3.7	2.9	3.6	44	209	48
Miguel AlmirÃ³n	PAR	Newcastle Utd	FW	29	33	23	1,937	3	0	1	2	0	4.5	4.5	2.6	78	78	162
Zeki Amdouni	SUI	Burnley	FW	22	34	27	1,953	4	1	1	2	0	5.8	5	1.4	63	49	93
Sofyan Amrabat	MAR	Manchester Utd	MF,DF	26	21	10	938	0	0	0	6	0	0.1	0.1	0.3	6	66	10
Joachim Andersen	DEN	Crystal Palace	DF	27	38	38	3,415	2	0	3	7	0	1.7	1.7	2.8	21	147	10
Mads Juel Andersе	DEN	Luton Town	DF	25	8	4	400	1	0	0	1	0	0.3	0.3	0	0	7	1
Elliot Anderson	ENG	Newcastle Utd	MF,FW	20	21	10	1,027	0	0	2	3	0	1.5	1.5	1.9	37	57	70
Jaidon Anthony	ENG	Bournemouth	FW	23	3	2	152	0	0	0	1	0	0.1	0.1	0	2	8	9
Michail Antonio	JAM	West Ham	FW	33	26	21	1,695	6	0	2	6	0	5.8	5.8	0.9	48	22	104
Antony	BRA	Manchester Utd	FW	23	29	15	1,324	1	0	1	5	0	3.4	3.4	2.8	53	64	149
Oliver Arblaster	ENG	Sheffield Utd	MF	19	12	11	943	0	0	0	2	0	0	0	0.7	13	43	4
Cameron Archer	ENG	Sheffield Utd	FW,MF	22	29	21	1,832	4	0	1	1	0	5.5	5.5	2.8	33	15	113

Câu 2:

- **Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.**
- **Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file results2.csv, format như sau:**

		Median of Atttribute 1	Mean of Atttribute 1	Std of Atttribute 1	...
0	all				
1	Team 1				
...					
n	Team n				

- **Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.**
- **Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024.**

1. Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở từng chỉ số.

- Với bài tập này, em đã lưu code của mình vào file **'Bài_2_a.py'**
- Để thực hiện yêu cầu tìm top 3 cầu thủ có điểm số cao nhất và thấp nhất ở từng chỉ số, em thực hiện phân tích thống kê chi tiết các chỉ số của cầu thủ từ dữ liệu đã thu thập ở câu 1 qua file kết quả là '**results.csv**':
  - Đầu tiên em tiến hành khởi tạo môi trường và chuẩn bị dữ liệu:

```
import pandas as pd

# Đọc dữ liệu từ file CSV
data = pd.read_csv('results.csv')
```

  - Ý tưởng: sử dụng thư viện pandas để dễ dàng thao tác dữ liệu dưới dạng bảng cũng như việc sắp xếp hay lọc dữ liệu.
  - Mục đích: nhập thư viện cần thiết (pandas) và đọc dữ liệu từ file **'results.csv'** (file kết quả của câu 1) để sử dụng cho phân tích.

- Xác định các chỉ số cầu thủ cần phân tích: ngoại trừ cột tên, quốc tịch cũng như vị trí thi đấu ra thì tất cả các chỉ số còn lại ta đều lấy để tìm ra top 3 như yêu cầu của bài toán.

```
# Xác định các thuộc tính cần phân tích
attributes = [
    'Age', 'Matches', 'Starts', 'Minutes', 'non_penalty_goals', 'penalty_goals',
    'assists', 'yellow_cards', 'red_cards', 'xG', 'npxG', 'xAG', 'PrgC', 'PrgP', 'PrgR',
    'per90_Gls', 'per90_Ast', 'per90_G+A', 'per90_G-PK', 'per90_G+A-PK', 'per90_xG',
    'per90_xAG', 'per90_xG+xAG', 'per90_npxG', 'per90_npxG+xAG', 'GA', 'GA90', 'SoTA',
    'Saves', 'Save%', 'W', 'D', 'L', 'CS', 'CS%', 'PKatt', 'PKA', 'PKsv', 'PKm', 'GK_Save%',
    'Gls', 'Sh', 'SoT', 'SoT%', 'Sh/90', 'SoT/90', 'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK',
    'PKatt', 'xG_shooting', 'npxG_shooting', 'npxG/Sh', 'G-xG', 'np:G-xG', 'Pass_Cmp',
    'Pass_Att', 'Pass_Cmp%', 'TotDist', 'PrgDist', 'Short_Cmp', 'Short_Att', 'Short_Cmp%',
    'Medium_Cmp', 'Medium_Att', 'Medium_Cmp%', 'Long_Cmp', 'Long_Att', 'Long_Cmp%',
    'Ast', 'xAG', 'xA', 'A-xAG', 'KP', '1/3', 'PPA', 'CrsPA', 'PrgP', 'Pass_Live', 'Pass_Dead',
    'Pass_FK', 'Pass_TB', 'Pass_Sw', 'Pass_Crs', 'Pass_TI', 'Pass_CK', 'Corner_In', 'Corner_Out',
    'Corner_Str', 'Pass_Cmp_outcome', 'Pass_Off', 'Pass_Blocks', 'SCA', 'SCA90', 'SCA_type_Passlive',
    'SCA_type_Passdead', 'SCA_type_TO', 'SCA_type_Sh', 'SCA_type_Fld', 'SCA_type_Def', 'GCA',
    'GCA90', 'GCA_type_Passlive', 'GCA_type_Passdead', 'GCA_type_TO', 'GCA_type_Sh', 'GCA_type_Fld',
    'GCA_type_Def', 'Tkl', 'TklW', 'Def_3rd', 'Mid_3rd', 'Att_3rd', 'Challenges_Tkl', 'Challenges_Att',
    'Challenges_Tkl%', 'Challenges_Lost', 'Blocks', 'Blocks_Sh', 'Blocks_Pass', 'Blocks_Int',
    'Blocks_Tkl+Int', 'Blocks_Clr', 'Blocks_Err', 'Touches', 'Def_Pen', 'Def_3rd', 'Mid_3rd',
    'Att_3rd', 'Att_Pen', 'Live_Touches', 'Take_Att', 'Take_Succ', 'Take_Succ%', 'Take_Tkld',
    'Take_Tkld%', 'Carries', 'Carries_TotDist', 'Carries_ProDist', 'Carries_ProgC', 'Carries_1/3',
    'Carries_CPA', 'Carries_Mis', 'Carries_Dis', 'Rec', 'Rec_PrgR', 'PT_Starts', 'PT_Mn/Start',
    'PT_Compl', 'Subs', 'Subs_Mn/Sub', 'Subs_unSub', 'TS_PPM', 'TS_onG', 'TS_onGA', 'TSxG_onxG',
    'TSxG_onGA', 'Fls', 'Fld', 'Off', 'Crs', 'OG', 'Recov', 'Aerial_Won', 'Aerial_Lost', 'Aerial_Won%'
]
```

– Trước khi bắt đầu tìm kiếm top 3, em chuyển đổi các giá trị của từng chỉ số sang kiểu số để tiện cho việc so sánh.

```
# Chuyển đổi từng cột thành dạng số
for attribute in attributes:
    data[attribute] = pd.to_numeric(data[attribute], errors='coerce')
```

- "**errors='coerce'** là một tham số tuỳ chọn của hàm 'pd.to_numeric()' trong thư viện pandas. Nó giúp chúng ta chuyển đổi các giá trị không thể chuyển thành số (các giá trị N/a) thành NaN (Not a Number)

– Tìm top 3 cầu thủ có điểm số cao nhất và thấp nhất ở mỗi chỉ số:

- Khởi tạo 2 dictionary để lưu thông tin cầu thủ có điểm cao nhất và thấp nhất cho từng chỉ số

```
top_players = {}
bottom_players = {}
```

- Tiến hành lặp qua từng chỉ số với vòng lặp for:

```
# Lặp qua từng chỉ số trong danh sách attributes
for attribute in attributes:
```

- Sử dụng phương thức **'dropna()'** để loại bỏ tất cả các hàng trong DataFrame 'data' có giá trị NaN trong từng chỉ số mà ta đang xét và kết quả sẽ được lưu vào '**valid_data**' để giúp chúng ta chỉ tìm kiếm những dữ liệu hợp lệ:

```
# Loại bỏ các hàng có giá trị NaN trong chỉ số hiện tại
valid_data = data.dropna(subset=[attribute])
```

- Tìm kiếm 3 giá trị lớn nhất và nhỏ nhất của từng chỉ số, đồng thời xác định giá trị lớn nhất và nhỏ nhất của từng chỉ số:

```
# Tìm 3 giá trị lớn nhất và nhỏ nhất
top_3 = valid_data.nlargest(3, attribute)
bottom_3 = valid_data.nsmallest(3, attribute)

# Tìm giá trị tối đa và tối thiểu
max_value = top_3[attribute].max() if not top_3.empty else None
min_value = bottom_3[attribute].min() if not bottom_3.empty else None
```

- Lọc và lấy tất cả cầu thủ có điểm số bằng với điểm số lớn nhất, trong trường hợp số lượng cầu thủ có điểm số lớn nhất chưa thoả mãn yêu cầu là 3 người thì ta tiến hành tìm các giá trị tiếp theo để bổ sung vào danh sách:

```
# Lấy tất cả cầu thủ có điểm số cao hơn hoặc bằng với max_value
top_players[attribute] = valid_data[valid_data[attribute] >= max_value][['Name', attribute]].sort_values(by=attribute, ascending=False)

# Nếu không đủ 3 cầu thủ, tìm tiếp cho đến khi đủ
while len(top_players[attribute]) < 3:
    next_value = valid_data[valid_data[attribute] < top_players[attribute][attribute].min()][attribute].max()  # Tìm giá trị tiếp theo
    if pd.isna(next_value):  # Nếu không còn giá trị nào nữa
        break
    top_players[attribute] = valid_data[valid_data[attribute] >= next_value][['Name', attribute]].sort_values(by=attribute, ascending=False)

# Lấy tất cả cầu thủ có điểm số thấp hơn hoặc bằng với min_value
```

- Tương tự với lọc các cầu thủ có điểm số nhỏ nhất, ta cũng làm như vậy.

– Ghi kết quả ra file '**BAI_2_a.in**' : để tiện cho việc theo dõi cũng như dễ dàng xem và sử dụng thông tin khi cần thiết mà không phải chạy code nhiều lần, em xuất kết quả ra file **'BAI_2_a.in'** với định dạng UTF-8:

```
# Ghi kết quả ra file BAI_2_a.in
with open('BAI_2_a.in', 'w', encoding='utf-8') as file:
    for attribute in attributes:
        file.write(f"Top 3 cầu thủ có điểm cao nhất ở chỉ số {attribute}:\n")
        file.write(top_players[attribute].to_string(index=False))
        file.write("\n\n")


        file.write(f"Top 3 cầu thủ có điểm thấp nhất ở chỉ số {attribute}:\n")
        file.write(bottom_players[attribute].to_string(index=False))
        file.write("\n\n")

print("Kết quả đã được ghi vào file BAI_2_a.in")
```

– **Kết quả**: được lưu trong file '**BAI_2_a.in**':

```
BAI_2_a.in
1   Top 3 cầu thủ có điểm cao nhất ở chỉ số Age:
2                    Name  Age
3   Łukasz Fabiański    38
4        Thiago Silva    38
5        Ashley Young    38
6
7   Top 3 cầu thủ có điểm thấp nhất ở chỉ số Age:
8                    Name  Age
9          Leon Chiwome    17
10         Lewis Miley    17
11       Evan Ferguson    18
12  Facundo Buonanotte    18
```

2. ***Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội . Ghi kết quả ra file results2.csv***

– Với bài tập này em đã lưu code của mình vào file *'**Bài_2_b.py**'*

– Để thực hiện yêu cầu tìm trung vị, trung bình, độ lệch chuẩn của mỗi chỉ số, em thực hiện phân tích thống kê chi tiết các chỉ số của cầu thủ từ dữ liệu đã thu thập ở câu 1 qua file kết quả là '**results.csv**':

- Đầu tiên em tiến hành khởi tạo môi trường và chuẩn bị dữ liệu:

```
import pandas as pd

# Đọc dữ liệu từ file CSV
data = pd.read_csv('results.csv')
```

- Ý tưởng: sử dụng thư viện pandas để dễ dàng thao tác dữ liệu dưới dạng bảng, tạo ra một DataFrame có tên là '*data*'

- Mục đích: nhập thư viện cần thiết (pandas) và đọc dữ liệu từ file **'results.csv'** (file kết quả của câu 1) để sử dụng cho phân tích.

– Trước khi tính toán các giá trị, em tiến hành:

- Lọc các cột có giá trị là số: dòng code dưới đây chọn ra tất cả các cột có dữ liệu kiểu '**float**' hoặc '**int**' và lưu tên các cột này vào biến '**number_attributes**':

```
# Chọn các cột chỉ chứa dữ liệu số (các cột chỉ số cần phải tính)
number_attributes = data.select_dtypes(include=[float, int]).columns
```

- Khởi tạo một Dictionary để lưu kết quả: **'results'** được khởi tạo với một cột tên là '**Team**' bắt đầu với giá trị là '**all**' để ghi nhận các kết quả cho toàn bộ các chỉ số cho toàn giải đấu trước khi tính toán riêng cho từng đội.

```
# Khởi tạo từ điển kết quả với cột "Team"
results = {"Team": ["all"]}
```

– Tính toán trung vị, trung bình và độ lệch chuẩn cho từng chỉ số cho các cầu thủ trong toàn giải:

```
# Tính trung vị, trung bình và độ lệch chuẩn cho từng chỉ số cho cả giải đấu
for x in number_attributes:

    results[f"Median of {x}"] = [f"{data[x].median(skipna=True):.2f}"]
    results[f"Mean of {x}"] = [f"{data[x].mean(skipna=True):.2f}"]
    results[f"Std of {x}"] = [f"{data[x].std(skipna=True):.2f}"]
```

- Vòng lặp for tính toán trung vị, trung bình và độ lệch chuẩn của từng chỉ số trong 'number_attributes'.
- '**data[x].median(skipna=True)**': tính trung vị của cột x và bỏ qua các giá trị thiếu.
- '**data[x].mean(skipna=True)**': tính trung bình của cột x và bỏ qua các giá trị thiếu.
- '**data[x].std(skipna=True)**': tính độ lệch chuẩn của cột x và bỏ qua các giá trị thiếu.
- Mỗi kết quả tính được sẽ được lấy đến 2 chữ số thập phân và được thêm vào **'results'**.

– Tính toán trung vị, trung bình và độ lệch chuẩn cho từng chỉ số cho các cầu thủ trong từng đội bóng:

```
# Nhóm dữ liệu theo từng đội bóng và tính trung vị, trung bình, độ lệch chuẩn cho từng đội
for team, group in data.groupby("Team"):
    results["Team"].append(team)
    for x in number_attributes:
        results[f"Median of {x}"].append(f"{group[x].median(skipna=True):.2f}")
        results[f"Mean of {x}"].append(f"{group[x].mean(skipna=True):.2f}")
        results[f"Std of {x}"].append(f"{group[x].std(skipna=True):.2f}")
```

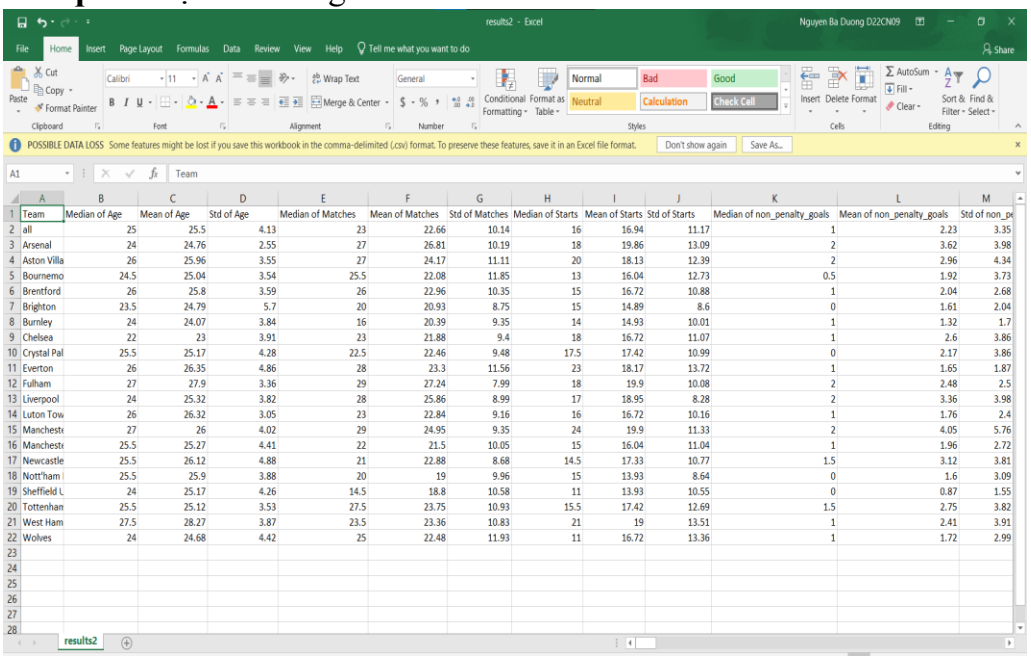
- Trong phần này, '*data*' sẽ được nhóm lại theo cột '**Team**' để ta dễ dàng tính toán các chỉ số theo từng đội bóng, và để làm được điều này em đã sử dụng '**groupby**'. Sau đó em thêm tên của từng đội bóng vào danh sách '**Team**'.
- Đối với mỗi chỉ số x trong '**number_attributes**', em lần lượt thực hiện việc tính toán giá trị trung vị, trung bình và độ lệch chuẩn của từng đội.
- Cuối cùng, kết quả được thêm vào '**results**'.

– Sau khi kết thúc quá trình tính toán, ta chuyển đổi kết quả thành DataFrame vào lưu kết quả vào file '**results2.csv**' :

```
# Chuyển kết quả thành DataFrame
results_df = pd.DataFrame(results)

# Lưu kết quả vào file 'results2.csv'
results_df.to_csv("results2.csv", index=False)
```

– **Kết quả**: được lưu trong file '**results2.csv**' :



Team	Median of Age	Mean of Age	Std of Age	Median of Matches	Mean of Matches	Std of Matches	Median of Starts	Mean of Starts	Std of Starts	Median of non_penalty_goals	Mean of non_penalty_goals	Std of non_pe
all	25	25.5	4.13	23	22.66	10.14	16	16.94	11.17	1	2.23	3.35
Arsenal	24	24.76	2.55	27	26.81	10.19	18	19.86	13.09	2	3.62	3.98
Aston Villa	26	25.96	3.55	27	24.17	11.11	20	18.13	12.39	2	2.96	4.34
Bournemo	24.5	25.04	3.54	25.5	22.08	11.85	13	16.04	12.73	0.5	1.92	3.73
Brentford	26	25.8	3.59	26	22.96	10.35	15	16.72	10.88	1	2.04	2.68
Brighton	23.5	24.79	5.7	20	20.93	8.75	15	14.89	8.6	0	1.61	2.04
Burnley	24	24.07	3.84	16	20.39	9.35	14	14.93	10.01	1	1.32	1.7
Chelsea	22	23	3.91	23	21.88	9.4	18	16.72	11.07	1	2.6	3.86
Crystal Pal	25.5	25.17	4.28	22.5	22.46	9.48	17.5	17.42	10.99	0	2.17	3.86
Everton	26	26.35	4.86	28	23.3	11.56	23	18.17	13.72	1	1.65	1.87
Fulham	27	27.9	3.36	29	27.24	7.99	18	19.9	10.08	2	2.48	2.5
Liverpool	24	25.32	3.82	28	25.86	8.99	17	18.95	8.28	2	3.36	3.98
Luton Tow	26	26.32	3.05	23	22.84	9.16	16	16.72	10.16	1	1.76	2.4
Manchest	27	26	4.02	29	24.95	9.35	24	19.9	11.33	2	4.05	5.76
Manchest	25.5	25.27	4.41	22	21.5	10.05	15	16.04	11.04	1	1.96	2.72
Newcastle	25.5	26.12	4.88	21	22.88	8.68	14.5	17.33	10.77	1.5	3.12	3.81
Nott'ham	25.5	25.9	3.88	20	19	9.96	15	13.93	8.64	0	1.6	3.09
Sheffield U	24	25.17	4.26	14.5	18.8	10.58	11	13.93	10.55	0	0.87	1.55
Tottenham	25.5	25.12	3.53	27.5	23.75	10.93	15.5	17.42	12.69	1.5	2.75	3.82
West Ham	27.5	28.27	3.87	23.5	23.36	10.83	21	19	13.51	1	2.41	3.91
Wolves	24	24.68	4.42	25	22.48	11.93	11	16.72	13.36	1	1.72	2.99

3. ***Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.***
   - Với bài tập này em đã lưu code của mình vào file '**Bài_2_c.py**'
   - Để thực hiện yêu cầu vẽ histogram phân bố của mỗi chỉ số, em thực hiện phân tích thống kê chi tiết các chỉ số của cầu thủ từ dữ liệu đã thu thập ở câu 1 qua file kết quả là '**results.csv'**, sau đó với mỗi nhóm chỉ số, em tiến hành vẽ biểu đồ histogram cho toàn giải đấu và cho riêng từng đội bóng rồi lưu vào các thư mục riêng.
   - **Import các thư viện cần thiết**:

```
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.ticker import MultipleLocator
from matplotlib.ticker import MaxNLocator
import os
```

     - Để có thể sử dụng dữ liệu từ file kết quả của câu 1 ta cần sử dụng thư viện '**pandas'** để dễ dàng thao tác dữ liệu dưới dạng bảng.
     - Bên cạnh đó, để vẽ biểu đồ ta cần dùng đến thư viện '**matplotlib'** . Không chỉ vậy, ta cần dùng đến module '**ticker'** của matplotlib giúp quản lí và tuỳ chỉnh các mốc (ticks) trên các trục của biểu đồ.
     - Cuối cùng là thư viện **OS** cho phép thực hiện các thao tác liên quan đến việc tạo, xoá file hay là thư mục.
   - **Đọc dữ liệu từ file csv kết quả của câu 1**:

```
# Tải dữ liệu lên
df = pd.read_csv('results.csv')
```

   - **Định nghĩa các nhóm chỉ số để dễ dàng vẽ biểu đồ**:

```
# Định nghĩa các  chỉ số cụ thể
categories = {
    "Age": ["Age"],
    "Playing Time": ["Matches", "Starts", "Minutes"],
    "Performance": ["non_penalty_goals", "penalty_goals", "assists", "yellow_cards", "red_cards"],
    "Expected": ["xG", "npxG", "xAG"],
    "Progression": ["PrgC", "PrgP", "PrgR"],
    "Per 90 minutes": ["per90_Gls", "per90_Ast", "per90_G+A", "per90_G-PK", "per90_G+A-PK", "per90_xG", "per90_xAG", "per90_xG+xAG", "per90_npxG", "per90_npxG+xAG"],
    "Goalkeeping Performance": ["GA", "GA90", "SoTA", "Saves", "Save%", "W", "D", "L", "CS", "CS%"],
    "Goalkeeping Penalties": ["PKatt", "PKA", "PKsv", "PKm", "Save%"],
    "Shooting Standard": ["Gls", "Sh", "SoT", "SoT%", "Sh/90", "SoT/90", "G/Sh", "G/SoT", "Dist", "FK", "PK", "PKatt"],
    "Shooting Expected": ["xG", "npxG", "npxG/Sh", "G-xG", "np:G-xG"],
    "Passing Total": ["Pass_Cmp", "Pass_Att", "Pass_Cmp%", "TotDist", "PrgDist"],
    "Passing Short": ["Short_Cmp", "Short_Att", "Short_Cmp%"],
    "Passing Medium": ["Medium_Cmp", "Medium_Att", "Medium_Cmp%"],
    "Passing Long": ["Long_Cmp", "Long_Att", "Long_Cmp%"],
    "Passing Expected": ["Ast", "xAG", "xA", "A-xAG", "KP", "1/3", "PPA", "CrsPA", "PrgP"],
    "Pass Types": ["Pass_Live", "Pass_Dead", "Pass_FK", "Pass_TB", "Pass_Sw", "Pass_Crs", "Pass_TI", "Pass_CK"],
    "Corners": ["Corner_In", "Corner_Out", "Corner_Str"],
    "Passing Outcomes": ["Pass_Cmp_outcome", "Pass_Off", "Pass_Blocks"],
    "Goal and Shot Creation": ["SCA", "SCA90", "SCA_type_Passlive", "SCA_type_Passdead", "SCA_type_TO", "SCA_type_Sh", "SCA_type_Fld", "SCA_type_Def", "GCA", "GCA90",
    "Defensive Actions": ["Tkl", "Tklw", "Def_3rd", "Mid_3rd", "Att_3rd", "Challenges_Tkl", "Challenges_Att", "Challenges_Tkl%", "Challenges_Lost", "Blocks", "Blocks_
    "Possession": ["Touches", "Def_Pen", "Def_3rd", "Mid_3rd", "Att_3rd", "Att_Pen", "Live_Touches", "Take_Att", "Take_Succ", "Take_Succ%", "Take_Tkld", "Take_Tkld%",
    "Playing Time": ["PT_Starts", "PT_Mn/Start", "PT_Compl", "Subs", "Subs_Mn/Sub", "Subs_unSub", "TS_PPM", "TS_onG", "TS_onGA", "TSxG_onxG", "TSxG_onGA"],
    "Miscellaneous Stats": ["Fls", "Fld", "Off", "Crs", "OG", "Recov", "Aerial_Won", "Aerial_Lost", "Aerial_Won%"]
}
```

- Khởi tạo một từ điển có tên là ‘categories’ để phân chia các chỉ số vào các nhóm cụ thể như trong hình. Điều này giúp chúng ta dễ tạo biểu đồ cho từng nhóm đồng thời thuận lợi cho việc quan sát và tìm kiếm.

– **Vẽ biểu đồ histogram cho mỗi chỉ số trong toàn giải đấu**:

- Tạo thư mục để lưu biểu đồ: để lưu biểu đồ của từng chỉ số cho toàn giải đấu, tiến hành tạo thư muc có tên là ‘histograms_by_all’:

```
# Tạo thư mục để lưu các biểu đồ histogram
output_all = 'histograms_by_all'
os.makedirs(output_all, exist_ok=True)
```

- Sử dụng vòng lặp for lặp qua từng nhóm chỉ số, đồng thời tạo thư mục tiêng cho mỗi nhóm chỉ số đó (category_dir):

```
# Tạo và lưu biểu đồ cho từng chỉ số
for category, stats in categories.items():
    category_dir = os.path.join(output_all, sanitize_filename(category))
    os.makedirs(category_dir, exist_ok=True)
```

- Vẽ biểu đồ: Với mỗi chỉ số ‘stat’ trong nhóm ‘stats’ ta tiến hành:

```
for stat in stats:
    if stat in df.columns:
        plt.figure(figsize=(12, 10))
        plt.hist(df[stat].dropna(), bins=20, alpha=0.6, color='blue')
```

- '**plt.figure(figsize=(12,10))**' : tạo khung hình để chứa biểu đồ, ở đây kích thước là 12x10 inches.
- '**plt.hist**()' :vẽ biểu đồ cho cột chỉ số với 20 bins và có màu xanh(blue) cùng với đó độ trong suốt của các cột trong histogram là 0.6.
- Thêm tiêu đề và nhãn cho biểu đồ:

```
plt.title(f'{stat} - {category}', fontsize=14)
plt.xlabel('Giá trị', fontsize=12)
plt.ylabel('Số lượng cầu thủ', fontsize=12)
```

- Thêm đường lưới ngang: '**plt.grid**()' thêm đường lưới trên biểu đồ. Ở đây ta chỉ thêm lưới cho trục y (axis='y') giúp dễ dàng so sánh chiều cao của các cột histogram.

```
    plt.grid(axis='y', alpha=0.7)
```

- Điều chỉnh mốc trên trục x: Với việc có tới hàng trăm cầu thủ và có tới hàng trăm giá trị khác nhau vì vậy việc giới hạn mốc trên trục x là cần thiết để giúp biểu đồ dễ đọc hơn. Ở đây, em đã để giới hạn tối đa là 15 mốc (ticks) đồng thời điều chỉnh kích thước phông chữ cho các mốc trên trục x và y dễ dàng cho việc quan sát.

```
# Giới hạn số tick trên trục x
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True, nbins=15))  # Chỉ hiển thị tối đa 15 tick
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
```

- Lưu và đóng biểu đồ: Sau khi vẽ xong ta tiến hành lưu biểu đồ dưới dạng file ảnh (.png), sau đó đóng biểu đồ sau khi lưu xong.

```
# Lưu biểu đồ dưới dạng ảnh riêng
plt.savefig(os.path.join(category_dir, f'{sanitize_filename(stat)}.png'))
plt.close()
```

– **Vẽ biểu đồ histogram cho từng chỉ số của từng đội**:

- Tạo thư mục chính có tên là '**histograms_by_team**' để lưu các biểu đồ của từng đội bóng đồng thời lấy danh sách tên các đội bóng từ cột 'Team' trong dataframe.

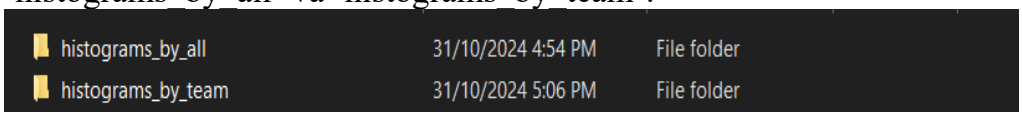
```
teams = df['Team'].unique()
team_output = 'histograms_by_team'
os.makedirs(team_output, exist_ok=True)
```

- Lặp qua từng đội bóng và tạo thư mục riêng cho từng đội.

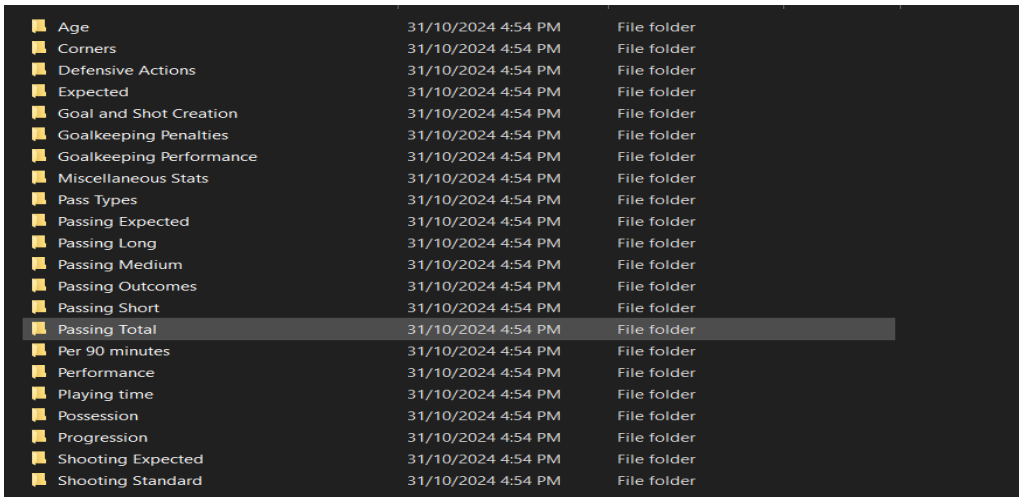
```
for team in teams:
    team_data = df[df['Team'] == team]
    team_dir = os.path.join(team_output, sanitize_filename(team))
    os.makedirs(team_dir, exist_ok=True)
```

- Vẽ biểu đồ: tương tự như phần vẽ biểu đồ cho toàn giải đấu.

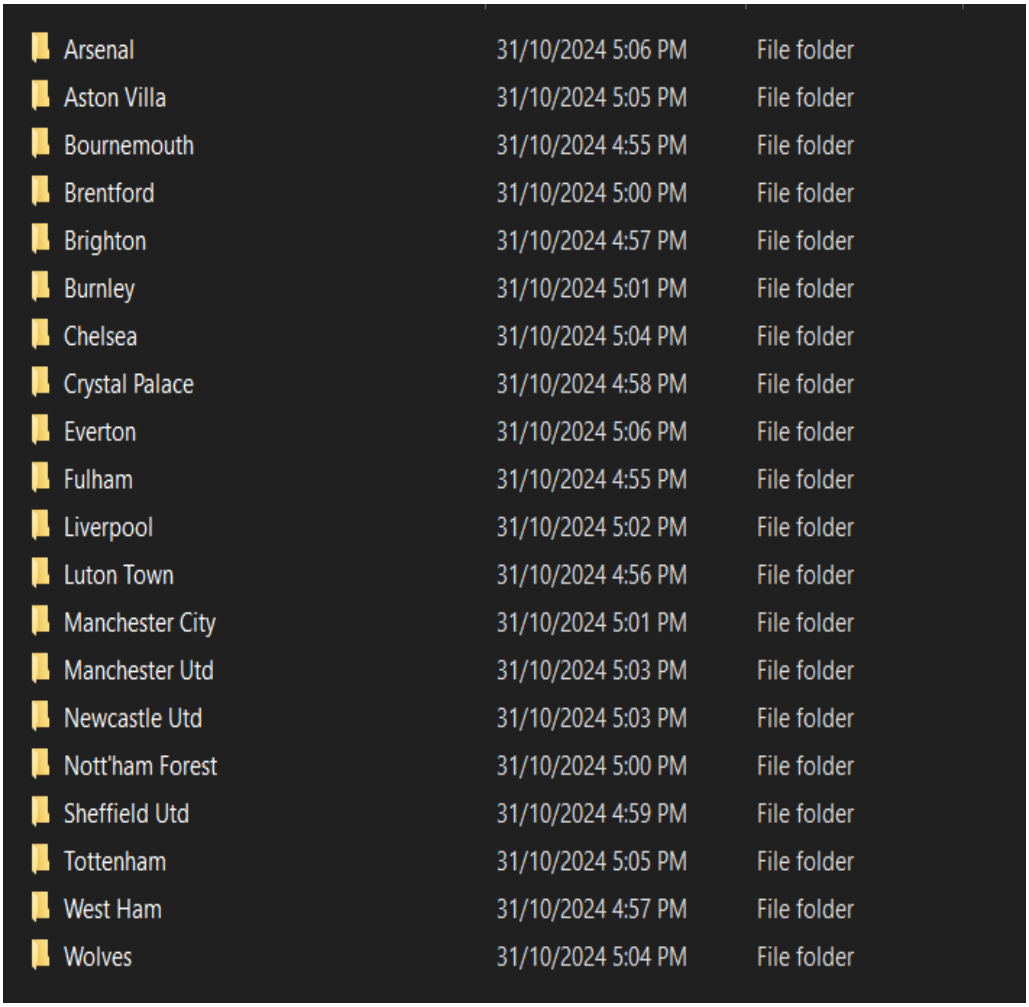
– **Kết quả**: các biểu đồ histogram đã được lưu vào 2 thư mục chính là ‘histograms_by_all’ và ‘histograms_by_team’:



- Trong thư mục ‘histograms_by_all’ : gồm các chỉ số của toàn giải đấu :



- Trong thư mục ‘histograms_by_team’: gồm các histogram của các đội bóng:



4. *Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024?*

4.1 Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số.

- Bài tập này em đã lưu code vào file '**Bài_2_d.py**' .
- **Đọc dữ liệu**:

```
import pandas as pd

# Đọc dữ liệu từ file results.csv
df = pd.read_csv('results.csv')
```

- Sử dụng thư viện '**pandas**' để xử lý và phân tích dữ liệu.

- Đọc dữ liệu từ file csv có tên là '**results.csv**' (file kết quả của câu 1) vào một DataFrame có tên là df.

– **Xử lý dữ liệu**:

```
#chuyển dữ liệu cột 'Minutes' sang dạng số
df['Minutes'] = df['Minutes'].str.replace(',', '').astype(float)
# ta nhóm dữ liệu theo cột 'Team'
teams = df.groupby('Team')
# khởi tạo dataframe rỗng để lưu kết quả
best_teams = pd.DataFrame(columns=['Attribute', 'Best Team', 'Value'])

# lọc ra các cột chứa dữ liệu dạng số
number_columns = df.select_dtypes(include=[float, int]).columns
```

- Do dữ liệu trong cột 'Minutes' đang không ở dạng số nên em chuyển về dạng số bằng cách loại bỏ dấu ','
- Nhóm dữ liệu theo cột 'Team' để có thể tính toán các chỉ số cho từng đội bóng.
- Khởi tạo một DataFrame rỗng để lưu kết quả và gồm có các cột là 'Attribute', 'Best Team', 'Value'.
- Đồng thời, ta lọc các cột chứa dữ liệu dạng số và bỏ qua các giá trị 'N/a'.

– **Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số**:

- Ta sẽ tính giá trị trung bình của mỗi đội cho từng chỉ số và tìm ra đội có giá trị trung bình cao nhất. Đó chính là đội có chỉ số điểm số cao nhất.

```
# Tìm đội có điểm số cao nhất cho từng chỉ số
for column in number_columns:
    # tính giá trị trung bình của mỗi đội cho cột đang xét
    best_team = teams[column].mean().idxmax()
    # trả về giá trị trung bình cao nhất của chỉ số
    best_value = teams[column].mean().max()

    # tạo một hàng dữ liệu mới
    row = pd.DataFrame({'Attribute': [column], 'Best Team': [best_team], 'Value': [best_value]})
    # thêm hàng này vào best_team
    best_teams = pd.concat([best_teams, row], ignore_index=True)
```

- Sử dụng vòng lặp for duyệt qua từng chỉ số, sau đó ta tính giá trị trung bình của từng đội bóng rồi tìm đội bóng có điểm số cao nhất là lưu tên đội bóng vào '**best_team**', đồng thời lưu giá trị cao nhất đó vào biến '**best_value'**.

- Tạo một hàng dữ liệu mới có tên là '**row**' với mục đích là tạo ra đầy đủ thông tin về đội bóng đó từ tên đội, chỉ số và giá trị cao nhất của chỉ số đó. Sau đó thêm hàng dữ liệu '**row**' vào DataFrame '**best_teams**'.

– **In kết quả ra màn hình và lưu kết quả vào file csv**:

```
#  in kết quả ra màn hình
print(best_teams)

# lưu kết quả vào file 'results3.csv'
best_teams.to_csv('results_bai_2d.csv', index=False)
```

- Chúng ta sẽ in kết quả ra màn hình để quan sát.
- Đồng thời sẽ lưu kết quả của bài tập vào file '**results_bai_2d.csv'**.

4.2 Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024

– Bài tập này em đã lưu code vào file '**Bài_2_e.py**'.

– **Ý tưởng**:

- Đối với một đội bóng, phong độ phải dựa trên 2 yếu tố đó là tấn công và phòng ngự. Một đội bóng có phong độ tốt thì cần phải có hàng công mạnh và một hàng thủ chắc chắn. Chính vì thế, để tìm ra đội bóng có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024, chúng ta cần phải dựa trên các chỉ số cụ thể trong mặt trận tấn công và phòng ngự.
- Phong độ của một đội bóng được tổng hợp bằng cách tính điểm trung bình trong các chỉ số về tấn công và phòng ngự rồi tổng hợp lại để cho ra một điểm số tổng quát về phong độ của đội bóng đó.

– **Đọc dữ liệu**:

```python
import pandas as pd

# Đọc dữ liệu từ file results.csv
df = pd.read_csv('results.csv')
teams = df.groupby('Team')
```

- Sử dụng thư viện '**pandas**' để xử lý và phân tích dữ liệu.
- Đọc dữ liệu từ file csv có tên là '**results.csv**' (file kết quả của câu 1) vào một DataFrame có tên là df.
- Nhóm dữ liệu theo tên đội bóng để thực hiện việc tính toán cho từng đội.

– **Xác định các chỉ số tấn công và phòng ngự cần thiết cho việc tính toán**:

```python
# Các chỉ số tấn công và phòng ngự
attack_metrics = ['non_penalty_goals', 'penalty_goals', 'assists', 'xG', 'npxG', 'xAG', 'per90_Gls', 'per90_Ast', 'per90_G+A']
defensive_metrics = ['Tkl', 'TklW', 'Def_3rd', 'Mid_3rd', 'Att_3rd', 'Challenges_Tkl', 'Blocks', 'Blocks_Sh', 'Blocks_Pass']
```

- Các chỉ số tấn công gồm có: **'non_penalty_goals'** (số bàn thắng không bao gồm penalty), **'penalty_goals'** (số bàn thắng bằng penalty), **'assists'** (kiến tạo), **'xG'** (bàn thắng dự kiến) , **'npxG'** (số bàn thắng dự kiến không phải phạt đền), **'xAG'** (số kiến tạo dự kiến) , **'per90_Gls'** (số bàn thắng ghi được trong 90 phút), **'per90_Ast'** (số kiến tạo dự kiến trong 90 phút), **'per90_G+A'** (số bàn thắng và kiến tạo dự kiến trong 90 phút). Các chỉ số này được lưu vào **'attack_metrics'**.
- Các chỉ số phòng ngự bao gồm: **'Tkl'** (số lượng cầu thủ bị hoá giải), **'TklW'** (giải quyết thành công và giành được quyền kiểm soát bóng), **'Def_3rd'** (giải quyết tình huống thành công trong 1/3 cuối sân), **'Mid_3rd '** (giải quyết tình huống thành công ở 1/3 giữa sân), **'Att_3rd'** (số lần tắc bóng thành công trong tấn công), **'Challenges_Tkl'** (số lượng cầu thủ dê bóng bị hoá giải), **'Blocks'** (số lần ngăn chặn thành công đường đi của bóng), **'Blocks_Sh'** (số lần ngăn chặn thành công cú sút của đối thủ), **'Blocks_Pass'** (số lần chặn thành công đường chuyền). Các chỉ số này được lưu vào **'defensive_metrics'**.

– **Tính điểm phong độ tấn công và phòng ngự**:

```
# Tính trung bình các chỉ số tấn công và phòng ngự cho từng đội
team_attack_performance = teams[attack_metrics].mean().mean(axis=1)
team_defensive_performance = teams[defensive_metrics].mean().mean(axis=1) / 100
```

- Điểm tấn công được tính bằng cách tính trung bình cộng của từng chỉ số tấn công cho mỗi đội bóng và tiếp tục lấy trung bình của các chỉ số đó.
- Điểm phòng ngự cũng được làm tương tự nhưng chia thêm cho 100 để chuẩn hoá dữ liệu phòng ngự với điểm tấn công, tạo điều kiện thuận lợi cho việc tính toán điểm phong độ chung.

– **Tính điểm phong độ chung**:

```
# Tạo bảng tổng hợp điểm phong độ
team_performance = pd.DataFrame({
    'Attack Performance': team_attack_performance,
    'Defensive Performance': team_defensive_performance
})

# Tính điểm phong độ chung bằng trung bình cộng của điểm tấn công và phòng ngự
team_performance['Overall Performance'] = team_performance.mean(axis=1)

# Sắp xếp theo thứ tự giảm dần của điểm phong độ chung
team_performance = team_performance.sort_values(by='Overall Performance', ascending=False)
```

- Tạo bảng để lưu kết quả của điểm chỉ số tấn công và phòng ngự.
- Tính điểm phong độ trung bằng trung bình của 2 điểm tấn công và phòng ngự và được lưu vào cột '**Overall Performance**'.
- Sau khi tính toán xong, ta sắp xếp theo thứ tự giảm dần của điểm phong độ chung để xác định đội bóng có phong độ tốt nhất.

– **Kết quả**:

```
# Lưu kết quả vào file results_bai_2e.csv
team_performance.to_csv('results_bai_2e.csv')

# In ra toàn bộ điểm phong độ tấn công, phòng ngự và điểm phong độ chung của từng đội bóng

print(team_performance)

# Tìm đội có phong độ cao nhất dựa trên điểm phong độ chung
best_team = team_performance['Overall Performance'].idxmax()
best_team_score = team_performance['Overall Performance'].max()

print(f"\nĐội bóng có phong độ tốt nhất của giải bóng đá Ngoại Hạng Anh mùa 2023-2024 là: {best_team} với điểm phong độ là: {best_team_score:.2f}")
```

- Kết quả được lưu vào file '**results_bai_2e.csv**'.

- Đồng thời ta tìm ra đội bóng có phong độ tốt nhất và điểm số phong độ của đội bóng đó và in ra màn hình.
- Đội bóng có phong độ tốt nhất mùa giải Ngoại Hạng Anh 2023-2024 là Manchester City. Đây cũng chính là đội bóng đã vô địch giải Ngoại Hạng Anh 2023-2024.

Câu 3:

- **Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.**
- **Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả.**
- **Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.**
- **Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ với đầu vào như sau:**
  - **python radarChartPlot.py --p1 <player Name 1> --p2 <player Name 2> --Attribute <att1,att2,...,att_n>**
  - **--p1: là tên cầu thủ thứ nhất**
  - **--p2: là tên cầu thủ thứ hai**
  - **--Attribute: là danh sách các chỉ số cần so sánh**

1. ***Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau. Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có nhận xét gì về kết quả.***

1.1 Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.

- Bài tập này được lưu vào file '**Bài_3_Kmeans.py'.**
- Để làm bài tập này, em sử dụng thuật toán K-means tìm số cụm tối ưu thông qua phương pháp Elbow.
- **Import các thư viện cần thiết**:

```
import pandas as pd
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
```

- Đầu tiên là thư viện **'pandas'** để đọc và xử lý dữ liệu từ file kết quả của câu 1 đó là '**results.csv**'.
- Tiếp theo là '**Kmeans' từ 'sklearn.cluster**' : Cung cấp thuật toán K-means để phân cụm dữ liệu.
- Bên cạnh đó là thư viện 'numpy' giúp xử lý dữ liệu ở phần sau.
- Cuối cùng là '**matplotlib.pyplot**' : Thư viện vẽ biểu đồ sẽ giúp chúng ta hiển thị biểu đồ Eblow để xác định số lượng cụm tối ưu.

– **Chuẩn bị dữ liệu**:

```
# Đọc dữ liệu
data = pd.read_csv("results.csv")

# Chuẩn bị dữ liệu (giả sử các cột từ 'Matches' đến cột cuối cùng)
fea = data.loc[:, 'Matches':]
for col in fea.columns:
    if fea[col].dtype == 'object':
        fea[col] = fea[col].str.replace(',', '').replace('N/a', np.nan).astype(float)
    else:
        fea[col] = fea[col].astype(float)
fea = fea.fillna(0)
```

- Đọc dữ liệu từ file csv: Ta tiến hành đọc dữ liệu từ file '**results.csv**' và lưu vào DataFrame có tên là 'data'.
- Lựa chọn các cột chỉ số: '**fea=data.loc[:, 'Matches':]**' sẽ chon tất cả các cột chỉ số từ cột '**Matches**' đến cột cuối cùng của 'data'.
- Xử lý dữ liệu của những cột dạng chuỗi hoặc N/a: Nếu có cột dữ liệu là object (chuỗi kí tự), ta tiến hành loại bỏ dấu phẩy hoặc thay thế các giá trị 'N/a' bằng '**np.nan**' để đánh dấu các giá trị này là NaN (giá trị thiếu). Sau đó ta chuyển toàn bộ các cột sang kiểu float.
- Xử lý giá trị NaN: '**fea=fea.fillna(0)**' thay thế bất kì giá trị thiếu (NaN) nào có trong 'fea' bằng giá trị 0, đảm bảo dữ liệu không còn giá trị trống.

– **Tìm số cụm tối ưu bằng phương pháp Elbow:**

```python
# Tìm k tối ưu bằng phương pháp Elbow
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(fea)
    wcss.append(kmeans.inertia_)
```

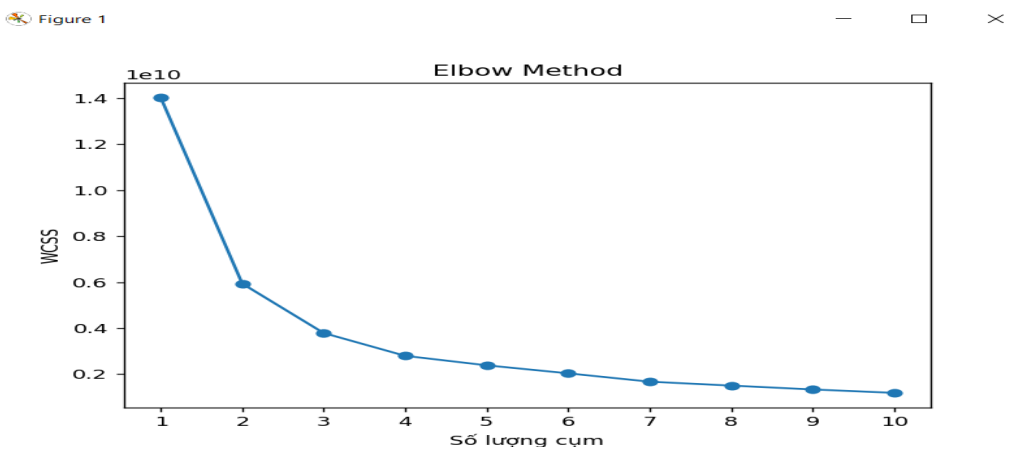
- Ta sử dụng vòng lặp để tính toán Within-Cluster Sum of Squares (WCSS) cho mỗi giá trị k từ 1 đến 10

– **Vẽ biểu đồ Elbow**: Biểu đồ Elbow được vẽ để xác định số cụm tối ưu:

```python
# Vẽ biểu đồ Elbow
plt.plot(range(1, 11), wcss,marker='o')
plt.title('Elbow Method')
plt.xlabel('Số lượng cụm')
plt.ylabel('WCSS')
plt.xticks(range(1, 11))  # Đảm bảo hiện tất cả các giá trị k trên trục x
plt.show()
```

- Trục X: số lượng cụm (k).
- Trục Y: WCSS
- Điểm 'khuỷu tay' (elbow) là điểm mà WCSS bắt đầu giảm với tốc độ chậm hơn nhiều. Và sau điểm này thì biểu đồ di chuyển gần như song song với trục X.

– **Xác định số cụm tối ưu**: Dựa vào biểu đồ Elbow ta vẽ được, ta có thể thấy với k=5 là giá trị gần như tối ưu nhất.



1.2 Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có nhận xét gì về kết quả.

- Dựa vào kết quả của biểu đồ Elbow, theo em ta nên phân loại cầu thủ thành 5 nhóm. Vì:
  - Bóng đá là môn thể thao mà mỗi cầu thủ có thể có nhiều vai trò và phong cách chơi khác nhau. Chính vì vậy, với 5 nhóm có thể phân loại cầu thủ dựa trên những chỉ số ta đã thu thập được về họ.
  - Đảm bảo tính chi tiết và đặc trưng cho từng nhóm: Phân cụm thành 5 nhóm cho phép mô tả chi tiết hơn về cầu thủ không chỉ ở các đặc trưng cơ bản như phòng ngự hay tán công mà còn nhiều yếu tố khác nữa.
  - Tính tương đồng giữa các cầu thủ trong nhóm sẽ cao hơn: Với 5 nhóm, các cầu thủ cùng một nhóm sẽ có mức độ tương đồng cao hơn, điều này giúp tăng cường sự chính xác trong khi đánh giá đặc điểm của cả nhóm.

- Nhận xét về kết quả:
  - Với kết quả phân cụm cầu thủ thành 5 nhóm dựa trên thuật toán K-means và biểu đồ Elbow theo em nghĩ là một sự phân loại hợp lí, giúp chia nhỏ dữ liệu thành các cụm có đặc điểm tương đồng cao.
  - Tuy nhiên, với việc có một số chỉ số ở dạng 'N/a' có thể ảnh hưởng và khiến việc phân nhóm trở nên không chính xác. Chính vì vậy, việc phân nhóm dựa trên kết quả của thuật toán K-means có thể chưa hoàn toàn đúng và áp dụng được trong các phân tích thực tế.

2. *Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.*

- Bài tập này được lưu vào file '**Bài_3_PCA.py'.**
- Dựa vào kết quả của bài K-means vừa rồi, ta tiến hành giảm số chiều dữ liệu xuống 2 chiều bằng thuật toán PCA.
- **Import các thư viện cần thiết**:

```
import pandas as pd
from sklearn.cluster import KMeans
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

- Đầu tiên là thư viện '**pandas**' để đọc và xử lý dữ liệu từ file kết quả của câu 1 đó là 'results.csv'.
- Tiếp theo là '**Kmeans**' từ '**sklearn.cluster**' : Cung cấp thuật toán K-means để phân cụm dữ liệu.
- Bên cạnh đó là thư viện '**numpy**' giúp xử lý dữ liệu ở phần sau.
- PCA từ '**sklearn.decomposition**' là kĩ thuật giúp giảm chiều dữ liệu để trực quan hoá phân cụm trong không gian 2 chiều.
- Cuối cùng là '**matplotlib.pyplot**' : Thư viện vẽ biểu đồ sẽ giúp chúng ta hiển thị biểu đồ Eblow để xác định số lượng cụm tối ưu.

– **Chuẩn bị dữ liệu**:

```
# Đọc dữ liệu từ file CSV
data = pd.read_csv("results.csv")

# Chuẩn bị dữ liệu (từ 'Matches' đến cột cuối cùng)
fea = data.loc[:, 'Matches':]
for col in fea.columns:
    if fea[col].dtype == 'object':
        fea[col] = fea[col].str.replace(',', '').replace('N/a', np.nan).astype(float)
    else:
        fea[col] = fea[col].astype(float)
fea = fea.fillna(0)
```

- Đọc dữ liệu từ file csv: Ta tiến hành đọc dữ liệu từ file '**results.csv**' và lưu vào DataFrame có tên là 'data'.
- Lựa chọn các cột chỉ số: '**fea=data.loc[:, 'Matches':]**' sẽ chon tất cả các cột chỉ số từ cột 'Matches' đến cột cuối cùng của 'data'.
- Xử lý dữ liệu của những cột dạng chuỗi hoặc N/a: Nếu có cột dữ liệu là object (chuỗi kí tự), ta tiến hành loại bỏ dấu phẩy hoặc thay thế các giá trị 'N/a' bằng '**np.nan**' để đánh dấu các giá trị này là NaN (giá trị thiếu). Sau đó ta chuyển toàn bộ các cột sang kiểu float.
- Xử lý giá trị NaN: **'fea=fea.fillna(0)'** thay thế bất kì giá trị thiếu (NaN) nào có trong '**fea**' bằng giá trị 0, đảm bảo dữ liệu không còn giá trị trống.

– **Phân cụm bằng K-means**:

```
# Phân cụm với k = 5
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10, random_state=0)
data['Cluster'] = kmeans.fit_predict(fea)
```

- Với kết quả từ bài trước là k=5, ta tiến hành khởi tạo **'k-means++'** giúp lựa chọn các tâm cụm ban đầu tốt hơn.
- **'max_iter=300'** : số lần lặp tối đ là 300 lần để thuật toán hội tụ.
- **'data['Cluster'] = kmeans.fit_predict(fea)'**: gán nhãn cụm cho từng cầu thủ trong data dựa trên phân cụm.

– **Giảm chiều dữ liệu bằng PCA**:

```
# Giảm số chiều dữ liệu bằng PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(fea)
data_pca = pd.DataFrame(data=principal_components, columns=['P1', 'P2'])
data_pca['Cluster'] = data['Cluster']
```

- '**PCA(n_component=2)**': ta tạo một đối tượng có tên là pca để giảm chiều dữ liệu xuống còn 2.
- Đồng thời ta giảm chiều của '**fea**' xuống còn 2 chiều qua câu lệnh '**pca.fit_transform(fea)**'.
- '**data_pca=pd.DataFrame(data=principal_components,columns=['P1','P2'])**' : chuyển đổi mảng thành DataFrame '**data_pca**' với hai cột là P1 và P2 đại diện cho 2 chiều.
- Cuối cùng ta thêm nhãn cụm vào '**data_pca**' để biết mỗi cầu thủ thuộc cụm nào.

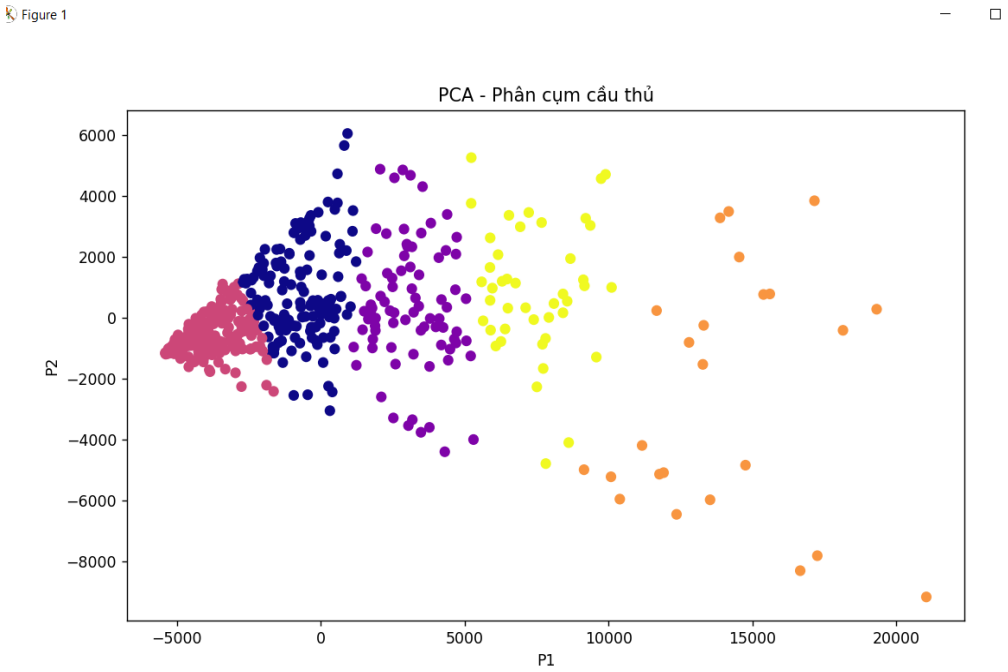
– **Vẽ biểu đồ phân cụm**:

```
# Vẽ biểu đồ phân cụm
plt.figure(figsize=(10,6))
plt.scatter(data_pca['P1'], data_pca['P2'], c=data_pca['Cluster'], cmap='plasma')
plt.title('PCA - Phân cụm cầu thủ')
plt.xlabel('P1')
plt.ylabel('P2')
plt.show()
```

- Ta tiến hành tạo một biểu đồ có kích thước 10x6 với giá trị của hai thành phần chính là '**data_pca['P1']**' và '**data_pca['P2']**' được sử dụng làm toạ độ X và Y trên biểu đồ.
- Màu của các điểm biểu diễn theo nhãn cụm từ 0 đến 4 ('c=data_pca['Cluster']'), đồng thời sử dụng bảng màu '**plasma**' để tô lên các điểm của từng cụm khác nhau.
- Đặt tiêu đề cho biểu đồ và cả tiêu đề cho trục X và trục Y.

- Cuối cùng là hiển thị biểu đồ phân tán, thể hiện các cụm cầu thủ trong không gian hai chiều.

– **Kết quả**:



3. ***Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ với đầu vào như sau:***
+ python radarChartPlot.py --p1 <player Name 1> --p2 <player Name 2> --Attribute
<att1,att2,...,att_n>
+ --p1: là tên cầu thủ thứ nhất
+ --p2: là tên cầu thủ thứ hai
+ --Attribute: là danh sách các chỉ số cần so sánh

– Bài tập này được em lưu code trong file '**radarChartPlot.py'.**
– Đoạn code trong file trên được dùng để giúp ta có thể vẽ biểu đồ radar nhằm so sánh các chỉ số giữa hai cầu thủ bất kì mà ta muốn so sánh. Với việc sử dụng biểu đồ radar giúp người dùng có thể dễ dàng so sánh 2 cầu thủ đó hơn.
– **Các thư viện sử dụng**:

```
import pandas as pd
import argparse
import matplotlib.pyplot as plt
from math import *
```

- Đầu tiên là thư viện ‘pandas’ giúp ta có thể đọc và phân tích dữ liệu từ file csv.
- ‘argparse’ cho phép chương trình nhận các tham số từ dòng lệnh, giúp người dùng có thể dễ dàng nhập tên cầu thủ và thuộc tính cần so sánh.
- Tiếp theo là thư viện ‘matplotlib.pyplot’ dùng để vẽ đồ thị, tạo ra các biểu đồ radar.

– **Hàm ‘radar_chart(player1, player2, attributes)’** :
- Mục đích: Dùng để thực hiện việc vẽ biểu đồ radar để so sánh giữa hai cầu thủ dựa trên các thuộc tính được cung cấp.
- Tham số gồm có : player1(cầu thủ thứ nhất), player2 (cầu thủ thứ hai), attributes (danh sách các thuộc tính).
- Đọc dữ liệu từ file ‘**results.csv**’ từ kết quả của câu 1.
- Lấy thông tin cầu thủ: Sử dụng câu lệnh lọc để tìm thông tin cầu thủ dựa trên tên và thuộc tính yêu cầu.

```
# Lấy thông tin cầu thủ
p1_data = data[data['Name'] == player1][attributes].iloc[0].values.flatten().tolist()
p2_data = data[data['Name'] == player2][attributes].iloc[0].values.flatten().tolist()
```

- Vẽ biểu đồ radar:

```
# Vẽ biểu đồ radar
angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)]
angles += angles[:1]

fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))

p1_data += p1_data[:1]
ax.plot(angles, p1_data, linewidth=1, linestyle='solid', label=player1)
ax.fill(angles, p1_data, 'blue', alpha=0.1)

p2_data += p2_data[:1]
ax.plot(angles, p2_data, linewidth=1, linestyle='solid', label=player2)
ax.fill(angles, p2_data, 'green', alpha=0.1)

plt.xticks(angles[:-1], labels)

ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))

plt.show()
```

Biểu đồ radar có dạng hình tròn với các trục tương ứng với từng thuộc tính. Chính vì vậy ra cần tính toán góc cho các trục bằng cách chia cả hình tròn cho số lượng thuộc tính.

Sau đó ta sử dụng matplotlib để tạo hình radar (polar plot).
Tiếp đến ta lần lượt vẽ dữ liệu cầu thủ thứ nhất và cầu thủ thứ hai. Tô màu cho vùng bên trong đường biểu diễn của từng cầu thủ để làm nổi bật.
Cuối cùng ta sử dụng '**plt.show()**' để hiển thị biểu đồ radar.

– **Chương trình chính**:

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--p1', type=str, required=True)
    parser.add_argument('--p2', type=str, required=True)
    parser.add_argument('--Attribute', type=str, required=True)

    args = parser.parse_args()

    player1 = args.p1
    player2 = args.p2
    attributes = args.Attribute.split(',')

    radar_chart(player1, player2, attributes)
```

- Mục đích: Phần này xử lý đầu vào từ dòng lệnh và gọi hàm để vẽ biểu đồ radar.
- Tham số nhận vào gồm: --p1 (tên cầu thủ thứ nhất), --p2 (tên cầu thủ thứ hai), --Attribute (danh sách các thuộc tính cần so sánh, cách nhau bởi dấu phẩy).
- Đầu tiên, ta thiết lập cho phép nhận các tham số từ dòng lệnh thông qua việc khởi tạo '**ArgumentParser**'.
- Tiếp theo, ta tiến hành phân tích đầu vào và gán giá trị cho các biến tương ứng.
- Cuối cùng ta tiến hành gọi hàm '**radar_chart**' với các tham số đã cho để tiến hành vẽ biểu đồ radar.

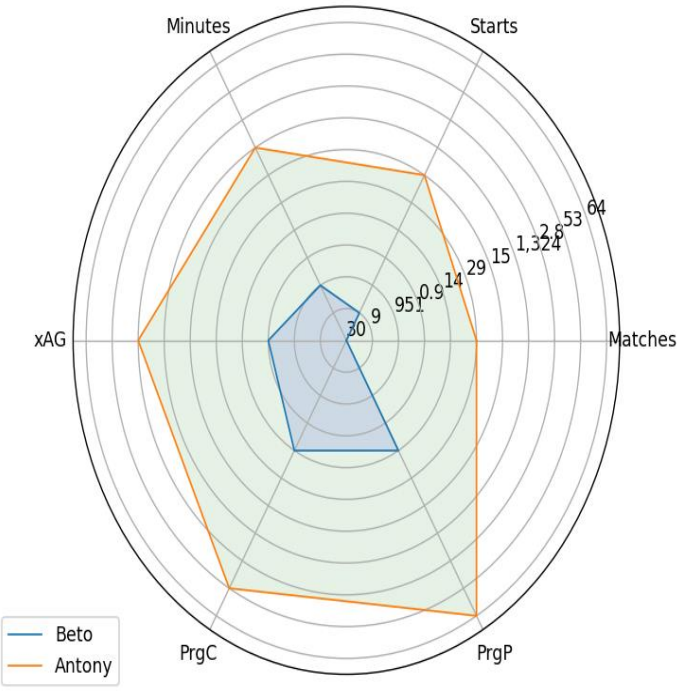
– **Cách chạy mã**:

- Để chạy được mã, ta cần mở terminal hoặc command prompt và điều hướng đến thư mục chứa file '**radarChartPlot.py**'.

- Sau đó ta chạy câu lệnh ‘python radarChartPlot.py --p1 <player Name 1> --p2 <player Name 2> --Attribute<att1,att2,...,att_n>’ để tiến hành vẽ biểu đồ radar so sánh hai cầu thủ.
- Ví dụ:

```
D:\admin\BTL_PYTHON>python radarChartPlot.py --p1 "Beto" --p2 "Antony" --Attribute "Matches,Starts,Minutes,xAG,PrgC,PrgP"
```

- Ở ví dụ trên ta có thể thấy cầu thủ thứ nhất là Beto và cầu thủ thứ hai là Antony, các thuộc tính cần so sánh là ‘Matches, Starts, Minutes, xAG, PrgC, PrgP’.
- Sau khi ta chạy câu lệnh thì ta sẽ được kết quả là một biểu đồ radar so sánh 2 cầu thủ này:



Câu 4: Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web https://www.footballtransfers.com. Đề xuất phương pháp định giá cầu thủ.

– **Giới thiệu**: Bài tập này em đã lưu code vào file '**Bài_4.py**'. Dữ liệu thu thập bao gồm có tên cầu thủ, đội bóng cũ, đội bóng mới chuyển tới và giá chuyển nhượng. Để làm được bài này, em sử dụng Selenium để tự động điều hướng trang web và dùng BeautifulSoup để phân tích HTML.
– **Các thư viện cần thiết**:

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
from time import sleep
```

- '**pandas**': Thư viện để xử lý và lưu trữ dữ liệu dưới dạng DataFrame.
- '**selenium**': Thư viện để tự động hoá trình duyệt web.
- '**BeautifulSoup**': Thư viện giúp chúng ta phân tích cú pháp HTML.
- Cuối cùng là '**webdriver_manager**' giúp ta tự động quản lý phiên bản driver cho trình duyệt Chrome.

– **Cấu hình Selenium**:

```
# Cấu hình Selenium để sử dụng Chrome
options = webdriver.ChromeOptions()

# Tạo driver với ChromeDriverManager
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=options)
```

- Cấu hình trình duyệt Chrome: Ta sử dụng '**webdriver.ChromeOptions()**'.
- Đồng thời ta khởi tạo driver bằng cách dùng '**ChromeDriverManager**' để có thể tự động cài phiên bản driver phù hợp với phiên bản Chrome.

– **Thu thập dữ liệu cầu thủ**:

```
# Khởi tạo danh sách để lưu dữ liệu cầu thủ
data = []

# Duyệt qua 18 trang (page 1 đến page 18)
for page in range(1, 19):
    if page == 1:
        url = "https://www.footballtransfers.com/us/transfers/confirmed/2023-2024/uk-premier-league"  # Trang
    else:
        url = f"https://www.footballtransfers.com/us/transfers/confirmed/2023-2024/uk-premier-league/{page}"

    driver.get(url)
    sleep(2)  # Đợi 2 giây để trang web tải xong
```

- Khởi tạo một danh sách dữ liệu có tên là data.
- Ta sử dụng một vòng lặp for để duyệt qua tổng cộng 18 trang của danh sách chuyển nhượng các cầu thủ.
- Do trang đầu tiên có URL khác với các trang còn lại nên ta để nó vào một trường hợp riêng.

– **Tiến hành phân tích HTML**: Sử dụng BeautifulSoup để phân tích cú pháp HTML.

```
# Lấy HTML
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
```

– **Tìm bảng dữ liệu cầu thủ**: Ta sử dụng phương thức find để tìm bảng có class tương ứng chứa thông tin chuyển nhượng cầu thủ.

```
# Tìm bảng dữ liệu cầu thủ
table = soup.find('table', {'class': 'table table-striped table-hover leaguetable mvp-table transfer-tabl
```

– **Lấy dữ liệu và trích xuất thông tin cầu thủ**:

```
# Tìm tất cả các hàng trong tbody (bỏ qua hàng tiêu đề trong thead)
rows = table.tbody.find_all('tr')

for row in rows:
    cols = row.find_all('td')
    if not cols:
        continue

    # Khởi tạo dictionary cho từng cầu thủ với giá trị mặc định là 'N/a'
    player = {'Name': 'N/a', 'Old_team': 'N/a', 'New_team': 'N/a', 'Price': 'N/a'}
```

- Ta lấy tất cả các hàng trong phần thân của ‘tbody’ để xử lý từng cầu thủ.
- Mỗi hàng sẽ được xử lý để lấy thông tin về cầu thủ. Khởi tạo một từ điển cho từng cầu thủ gồm có ‘Name, Old_team, New_team, Price’ và tất cả đều được khởi tạo với giá trị mặc định là ‘N/a’.

– **Tìm kiếm tên cầu thủ**:

```
name_span = cols[0].find('span', class_='d-none')
if name_span:
    player['Name'] = name_span.text.strip()
```

- Ta tìm kiếm tên cầu thủ trong cột đầu tiên của bảng dữ liệu và tên cầu thủ nằm trong thẻ span với class là ‘d-none’.

– **Tìm tên đội bóng cũ và tên đội bóng mới**:

```
# Tìm đội bóng cũ của cầu thủ và đội mới mà cầu thủ chuyển nhượng đến
old_team_div = cols[1].find('div', class_='transfer-club transfer-club--from')
if old_team_div:
    old_team_name = old_team_div.find('div', class_='transfer-club__name')
    player['Old_team'] = old_team_name.text.strip() if old_team_name else 'N/a'

new_team_div = cols[1].find('div', class_='transfer-club transfer-club--to')
if new_team_div:
    new_team_name = new_team_div.find('div', class_='transfer-club__name')
    player['New_team'] = new_team_name.text.strip() if new_team_name else 'N/a'
```

- Tìm tên đội bóng cũ: Tên đội bóng cũ nằm trong cột thứ hai và trong thẻ div có class là ‘transfer-club transfer-club—from’.
- Tìm tên đội bóng mới: Tên đội bóng mới nằm trong cùng cột thứ hai với tên đội bóng cũ và nằm trong thẻ div có class là ‘transfer-club transfer-club—to’.

– **Tìm kiếm giá chuyển nhượng**:

```
# Tìm giá trị 'Price' từ thẻ span trong cols[3]
price_span = cols[3].find('span')
if price_span:
    price_text = price_span.text.strip()
# Nếu giá trị là "Free" thì giữ nguyên. Nếu có tiền chuyển nhượng thì xoá ký hiệu
    if "Free" in price_text:
        player['Price'] = price_text
    else:
    # Xoá ký hiệu tiền tệ, giữ lại phần số và chữ cái (M,K)
        player['Price'] = price_text.replace('€', '').replace('$', '').strip()
else:
    player['Price'] = 'N/a'
```

- Ta tìm kiếm giá trị Price trong cột thứ tư của thẻ span.

- Xử lý giá trị: Nếu giá trị chứa từ ‘Free’ thì gán giá trị đó cho khoá Price. Còn không, ta sẽ loại bỏ ký hiệu tiền tệ và lưu lại số liệu cũng như là loại bỏ khoảng trắng dư thừa.

– **Thêm cầu thủ vào danh sách dữ liệu**:

```
# Thêm cầu thủ vào danh sách
data.append(player)
```

– **Chuyển đổi danh sách dữ liệu thành DataFrame và lưu kết quả**:

```
# Chuyển đổi danh sách dữ liệu thành DataFrame
df = pd.DataFrame(data)

# In kết quả
print(df)

# Lưu DataFrame vào file CSV
df.to_csv('Bài_4.csv', index=False)
```

- Ta chuyển đổi danh sách dữ liệu thành một DataFrame có tên là ‘df’.
- Sau đó, ta lưu DataFrame vào file csv có tên là ‘Bài_4.csv’.

– **Kết quả**:

	A	B	C	D
1	Name	Old_team	New_team	Price
2	Demi Akarakiri	Arsenal	Everton	Free
3	Kain Ryan	Brighton	Coventry	Free
4	Nicksoen Gomis	Sheff Utd	Toronto	Free
5	Richie Laryea	Nottingham	Toronto	0.7M
6	Bruno JordÃ£o	Wolverhampton	Radomiak	Free
7	Charlie Wiggett	Newcastle Utd.	Sligo	Free
8	Serge Aurier	Nottingham	Galatasaray	0.1M
9	Alfie Harrison	Man City	Newcastle Utd.	1M
10	Bertrand TraorÃ©	Aston Villa	Villarreal	Free
11	Matz Sels	Strasbourg	Nottingham	8M
12	John Fleck	Sheff Utd	Blackburn	Free
13	Thierry Small	Southampton	Charlton Athletic	Free
14	Adam Wharton	Blackburn	C. Palace	21.1M
15	Joe Gauci	Adelaide	Aston Villa	1.5M
16	Theo Corbeanu	Wolverhampton	Granada	1.5M
17	Morgan Rogers	Middlesbrough	Aston Villa	9.4M
18	Pablo Fornals	West Ham United	Real Betis	8M
19	Isak Hansen-AarÃ¸en	Man Utd	Werder	0.3M
20	RÃ³nar Alex RÃ³narsson	Arsenal	Kobenhavn	Free
21	Miguel Azeez	Arsenal	At. Baleares	Free
22	Hugo Lloris	Tottenham	LAFC	Free
23	Zack Steffen	Man City	Colorado	Free
24	Daniel MuÃ±oz	Genk	C. Palace	8M
25	Daiki Hashioka	Sint-Truiden	Luton	2M
26	Jonny Otto	Wolverhampton	PAOK	Free
27	HÃ¡kon Rafn Valdimarsson	Elfsborg	Brentford	3M
28	Ivo Grbic	AtlÃ©tico	Sheff Utd	2.5M

TÀI LIỆU THAM KHẢO

- Selenium:
  - https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test
  - https://anhtester.com/blog/cac-lenh-truy-van-trong-selenium-webdriver-python-b330.html
  - https://www.youtube.com/watch?v=d4dnEL9tpMc
- Histogram:
  - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
  - https://www.youtube.com/watch?v=mUktHrouJ_o
  - https://www.youtube.com/watch?v=VH2JgqlN2so
- K-means:
  - https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/
  - https://www.youtube.com/watch?v=-5EODZFlOiA
- PCA:
  - https://www.youtube.com/watch?v=8klqIM9UvAc
  - https://www.youtube.com/watch?v=QdBy02ExhGI
- Radar:
  - https://www.youtube.com/watch?v=bERw8XfaQaQ
  - https://matplotlib.org/stable/gallery/specialty_plots/radar_chart.html
  - https://plotly.com/python/radar-chart/