HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH PYTHON

NHÓM 11

BÁO CÁO BÀI TẬP LỚN PYTHON

NGUYỄN VIẾT TUẤN ANH – B22DCCN038

Hà Nội – 2024

Mục lục

I. Công nghệ và thuật toán sử dụng.

1. Công nghệ và các thư viện sử dụng

Selenium là một bộ công cụ tự động hóa trình duyệt web, cho phép người dùng tự động hóa các thao tác trên trình duyệt, như nhấp chuột, điền biểu mẫu và thu thập dữ liệu từ các trang web.

Chức năng:

- Tự động hóa kiểm tra giao diện người dùng.
- Lấy dữ liệu từ các trang web mà không có API.
- Hỗ trợ đa trình duyệt, bao gồm Chrome, Firefox, và Safari.
- Chạy JavaScript và xử lý phần tử động: Selenium có thể thực thi mã JavaScript trên trang web và xử lý các phần tử động được tải qua AJAX.

Beautifulsoup: là một thư viện Python dùng để phân tích cú pháp HTML và XML, giúp dễ dàng trích xuất dữ liệu từ các trang web. Nó rất hữu ích khi bạn cần lấy dữ liệu từ trang web mà không cần phải điều khiển trình duyệt (như Selenium). BeautifulSoup đặc biệt phù hợp với các trang tĩnh hoặc khi đã có mã HTML của trang cần xử lý.

Chức năng:

- Tìm kiếm và lấy dữ liệu từ các tài liệu HTML/XML.
- Tích hợp tốt với các thư viện phân tích cú pháp khác như html.parser (mặc định), lxml, hoặc html5lib để phân tích cú pháp nhanh hơn và chính xác hơn.
- Cho phép tìm kiếm các thẻ HTML bằng nhiều cách như tên thẻ, id, class, hoặc các thuộc tính tùy ý khác.
- Cung cấp các phương thức như find(), find_all(), và select() để tìm và lấy dữ liệu một cách hiệu quả.

Pandas: là một thư viện phân tích và xử lý dữ liệu mạnh mẽ trong Python, rất phổ biến cho các tác vụ liên quan đến dữ liệu trong khoa học dữ liệu và học máy. Với Pandas, bạn có thể dễ dàng thao tác với dữ liệu dạng bảng, tính toán, lọc, và chuyển đổi dữ liệu thành các định dạng khác.

Đặc điểm nổi bật của Pandas:

- Cấu trúc dữ liệu linh hoạt:
    - **DataFrame**: Cấu trúc dữ liệu hai chiều (giống như bảng) có thể chứa nhiều kiểu dữ liệu khác nhau.

    - **Series**: Cấu trúc dữ liệu một chiều, thường được dùng để biểu diễn một cột hoặc một hàng của DataFrame.
- Dễ dàng xử lý dữ liệu: Pandas cung cấp các hàm tiện lợi để lọc, gom nhóm, tính toán thống kê, xử lý dữ liệu thiếu, và chuyển đổi dữ liệu.
- Đọc và ghi dữ liệu:
    - Pandas hỗ trợ nhiều định dạng dữ liệu, bao gồm CSV, Excel, SQL, JSON, HTML, và các định dạng khác.
    - Cho phép lưu và tải dữ liệu từ các nguồn khác nhau, phù hợp cho các tác vụ trích xuất dữ liệu.
- Chuyển đổi dữ liệu nhanh chóng: Dễ dàng thay đổi định dạng dữ liệu, chuyển từ bảng thành mảng, từ mảng thành bảng, gộp hoặc chia nhỏ dữ liệu.

Matplotlib: là thư viện trực quan hóa dữ liệu mạnh mẽ trong Python, thường được sử dụng để tạo các loại biểu đồ khác nhau như biểu đồ đường, cột, tròn, tán xạ, và nhiều loại khác. Thư viện này rất hữu ích cho việc trình bày và phân tích dữ liệu, giúp bạn dễ dàng nhận ra các xu hướng và mẫu trong dữ liệu.

Đặc điểm nổi bật của Pandas:

- Đa dạng kiểu biểu đồ: Matplotlib cung cấp nhiều loại biểu đồ như biểu đồ đường, biểu đồ cột, biểu đồ tròn, biểu đồ tán xạ, và các biểu đồ dạng 3D.
- Tùy chỉnh linh hoạt: có thể tùy chỉnh màu sắc, kiểu đường, kiểu điểm, trục tọa độ, nhãn, tiêu đề, chú thích và nhiều yếu tố khác.
- Hoạt động tốt với các thư viện khác: Matplotlib có thể dễ dàng tích hợp với các thư viện như Pandas và NumPy để trực quan hóa dữ liệu từ các DataFrame hoặc mảng số.

2. Các thuật toán và thư viện sử dụng.

Elbow: Phương pháp Elbow (khuỷu tay) là một kỹ thuật thường được sử dụng trong KMeans để xác định số lượng cụm (k) tối ưu. Ý tưởng là chạy thuật toán KMeans với các giá trị khác nhau của kkk và sau đó tính toán tổng lỗi bình phương nội cụm (Within-Cluster Sum of Squares - WCSS) cho mỗi giá trị kkk. Tổng lỗi bình phương nội cụm thể hiện khoảng cách từ các điểm trong cụm đến trung tâm của cụm đó.

Điểm *khủy tay* là điểm mà ở đó tốc độ suy giảm của *hàm biến dạng* sẽ thay đổi nhiều nhất.

Phương pháp Elbow là một phương pháp thường được sử dụng để lựa chọn số lượng cụm phân chia hợp lý dựa trên biểu đồ.

Tuy nhiên có một số trường hợp chúng ta sẽ không dễ dàng phát hiện vị trí của Elbow, đặc biệt là đối với những bộ dữ liệu mà quy luật phân cụm không thực sự dễ dàng được phát hiện.

PCA: hay còn gọi là phân tích thành phần chính, là một phương pháp giảm chiều dữ liệu rất phổ biến trong học máy và thống kê. PCA giúp bạn giảm số chiều của dữ liệu nhiều chiều xuống ít chiều hơn trong khi vẫn giữ lại phần lớn thông tin quan trọng, từ đó đơn giản hóa dữ liệu và giúp việc phân tích dễ dàng hơn.

Thuật toán và cách hoạt động:

Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\mathbf{x} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

Trừ mỗi điểm dữ liệu với vector kỳ vọng của toàn bộ dữ liệu.

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \mathbf{x}$$

Tính ma trận hiệp phương sai.

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

Tính các giá trị riêng và vector riêng có norm bằng 1 của ma trận hiệp phương sai, sắp xếp theo thứ tự giảm dần của giá trị riêng.

Chọn K vector riêng ứng với K giá trị riêng lớn nhất để xây dựng ma trận U_K có các cột tạo thành một hệ trực giao. K vectors này được gọi là các thành phần chính, tạo thành một không gian con gần với phân bố của dữ liệu ban đầu.

Chiếu dữ liệu ban đầu theo không gian con tìm được.

Dữ liệu mới là tọa độ của các điểm dữ liệu trên không gian mới.

$$\mathbf{Z} = \mathbf{U}_K^T\hat{\mathbf{X}}$$

KMeans: là thuật toán phân cụm dữ liệu được đặc trưng bởi một *tâm* (*centroid*). *tâm* là điểm đại diện nhất cho một cụm và có giá trị bằng

trung bình của toàn bộ các quan sát nằm trong cụm. Chúng ta sẽ dựa vào khoảng cách từ mỗi quan sát tới các *tâm* để xác định nhãn cho chúng trùng thuộc về *tâm* gần nhất.

Cụ thể các bước của thuật toán k-Means được tóm tắt như sau:

Khởi tạo ngẫu nhiên k tâm cụm k1, k2, k3…

Lặp lại quá trình cập nhật tâm cụm cho tới khi dừng:

- Xác định nhãn cho từng điểm dữ liệu ci dựa vào khoảng cách tới từng tâm cụm:

$$c_i = \arg\min_j \|\mathbf{x}_i - \mu_j\|_2^2$$

- Tính toán lại tâm cho từng cụm theo trung bình của toàn bộ các điểm dữ liệu trong một cụm:

$$\mu_j := \frac{\sum_{i=1}^{n} \mathbf{1}(c_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} \mathbf{1}(c_i = j)}$$

II. Thực hiện

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.

```
def openDriver(url):
    driver.get(url)
    time.sleep(5)
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    return soup
```

Hàm OpenDriver nhận vào một url được khai bảo ở trên như: urlRoot, goalKeepingUrl, shootingUrl, passingUrl, passTypeUrl, goalAndShotUrl, defensiveUrl, possessionUrl, playingTimeUrl, miscellaneousUrl. Sẽ mở các url bằng Selenium, tải trang, và trả về mã HTML của trang dưới dạng một đối tượng BeautifulSoup.

```
def findInStats(texts, s):
    player = []
```

```
        td = texts.find_all('td', attrs = {"data-stat":s})
        for x in td:
            if len(str(x)) != 0:
                player.append(x.text.strip())
        return player
```

Hàm findInStats nhận vào một hai tham số texts và s. Texts là dữ liệu HTML bảng số liệu thống kê tương ứng với từng url ở trên. S là một chuỗi kí tự thuộc tính cột cần tìm trong bảng. findInStats sẽ tìm toàn bộ dữ liệu của cột có thuộc tính là S và sẽ trả về một danh sách dữ liệu của thuộc tính đó ở dạng chuỗi ký tự.

```
res['player'] = findInStats(texts, 'player')
res['nationality'] = findInStats(texts, 'nationality')
res['team'] = findInStats(texts, 'team')
res['position'] = findInStats(texts, 'position')
res['age'] = findInStats(texts, 'age')
```

Res, res1, res2,.., res9 là các Dictionary dùng để lưu dữ liệu lấy từ từng url. Với khóa là tên thuộc tính hay tên cột và giá trị là danh sách các giá trị của thuộc tính đó.

```
df_stats = pd.DataFrame(res)
df_goalKeeper = pd.DataFrame(res1)
df_shooting = pd.DataFrame(res2)
df_passing = pd.DataFrame(res3)
df_typePass = pd.DataFrame(res4)
df_goalAndShot = pd.DataFrame(res5)
df_defensive = pd.DataFrame(res6)
df_possession = pd.DataFrame(res7)
df_playingTime = pd.DataFrame(res8)
df_miscellaneous = pd.DataFrame(res9)
```

Sử dụng Pandas để tạo mới các dataFrame từ các dictionary ở trên lần lượt có chín dataFrame: df_stats, df_goalKeeper, df_shooting, df_passing, df_typePass, df_goalAndShot, df_defensive, df_possession, df_playingTime, df_miscellaneous.

```
df_result = df_stats.merge(df_goalKeeper, on = ['player','team'], how='outer')
df_result =df_result.merge(df_shooting, on=['player','team'], how = 'outer')
df_result =df_result.merge(df_passing, on=['player','team'], how = 'outer')
df_result = df_result.merge( df_typePass, on=['player','team'], how = 'outer')
df_result = df_result.merge( df_goalAndShot, on=['player','team'], how = 'outer')
df_result = df_result.merge( df_defensive, on=['player','team'], how = 'outer')
df_result = df_result.merge( df_possession, on=['player','team'], how = 'outer')
df_result = df_result.merge(df_playingTime, on=['player','team'], how = 'outer')
df_result = df_result.merge( df_miscellaneous, on=['player','team'], how = 'outer')
```

Hợp nhất các dataframe ở trên thành một dataframe là df_result. Dòng đầu tiên sẽ hợp nhất df_stat và df_goalKeeper dựa trên hai cột là cột player và cột team. Sử dụng how với giá trị outer có nghĩa là một giá trị của cặp player và team trong hai dataframe sẽ được giữ lại, ngay cả khi một số giá trị chỉ có mặt ở một trong hai dataframe.

Tiếp tục hợp nhất các dataframe tiếp theo vào df_result. Nếu cầu thủ ko có giá trị trong một dataframe thì các cột tương ứng sẽ có giá trị NaN. Df_result sẽ là dataframe tổng hợp chứa tất cả thông tin từ các dataframe ban đầu. Mỗi hàng trong df_result đại diện cho một cầu thủ và mỗi cột là một số liệu khác nhau thuộc về cầu thủ này.

Các đoạn code tiếp theo sẽ thực hiện việc chuẩn bị và xử lý dữ liệu trước khi lưu vào file csv.

```
df_result['minutes'] = df_result['minutes'].str.replace(',','')
df_result['minutes'] = pd.to_numeric(df_result['minutes'], errors='coerce')
df_result = df_result[df_result['minutes'] >= 90]
```

Chức năng của đoạn code trên là lọc ra các cầu thủ có số phút thi đấu lớn hơn hoặc bằng chin mươi phút. Dòng đầu tiên giúp chuẩn hóa dữ liệu của cột minutes về dạng mỗi chuỗi ký tự và không có dấu phẩy. Dòng tiếp theo làm

chức năng chuyển đổi cột minutes sang kiểu số (numeric) với errors có giá trị là coerce. Nếu có giá trị nào không thể chuyển được nó sẽ thành NaN. Cuối cùng lọc ra các cầu thủ có số phút thi đấu lớn hơn hoặc bằng chin mươi phút.

```
df_result.fillna('n/a', inplace=True)

df_result.replace('', 'n/a', inplace=True)
```

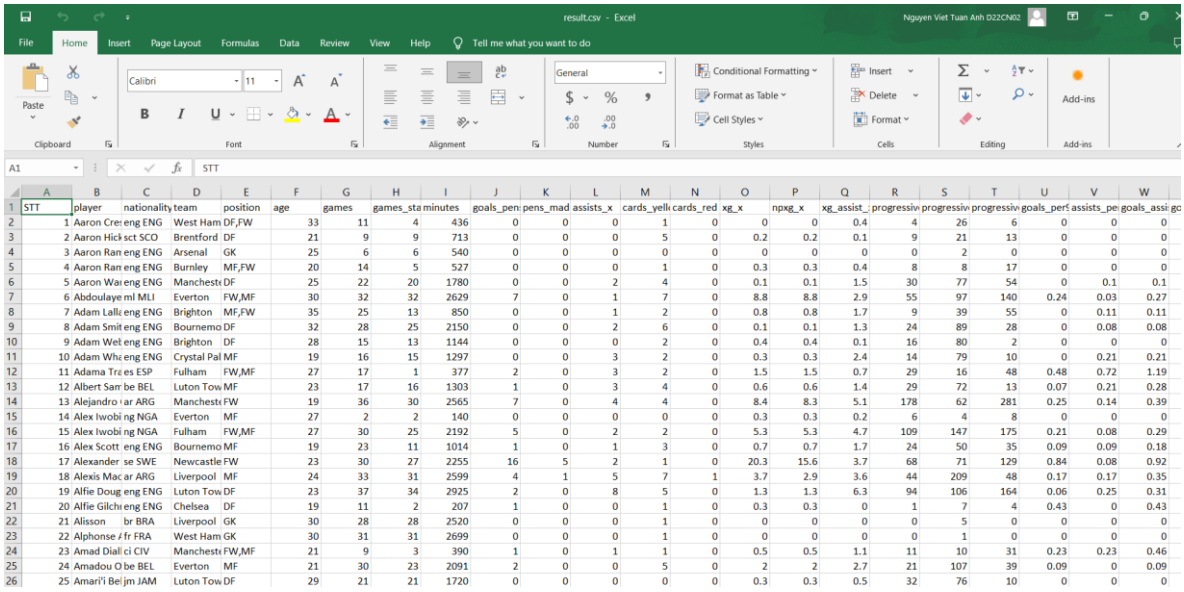
Hàm fillna sẽ điền gái trị ‘n/a’ và các ô có giá trị NaN. Sau đó hàm replace sẽ thay thế các ô trống mà không phải NaN thành n/a.

```
df_result = df_result.sort_values(by=['player', 'age'], ascending=[True, False])

df_result.insert(0, 'STT', range(1, len(df_result) + 1))

df_result.to_csv('C:/ result.csv', index=False)
```

Đây là bước sắp xếp và chuẩn hóa cuối cùng trước khi lưu thành csv. Hàm sort_values sẽ có hai tham số là by: cột dữ liệu muốn sắp xếp và ascending: True là sắp xếp tăng dần và False là sắp xếp giảm dần.

Sau khi hợp nhất các dataframe thì số thứ tự khi chuyển qua file csv ko đúng trật tự. Hàm insert sẽ có chức nằng thêm một cột là STT vào dataframe bắt đầu từ vị trí 0 và có giá trị tăng dần bắt đầu từ 1.

Cuối cùng lưu df_result vào file csv them đường dẫn trong hàm to_csv và thêm tham số index = false để bỏ cột chỉ mục khi lưu file.



STT	player	nationality	team	position	age	games	games_sta	minutes	goals_pen	pens_mad	assists_x	cards_yell	cards_red	xg_x	npxg_x	xg_assist_	progressiv	progressiv	progressiv	goals_per	assists_pe	goals_assi
1	Aaron Cre	eng ENG	West Ham	DF,FW	33	11	4	436	0	0	0	1	0	0	0	0.4	4	26	6	0	0	0
2	Aaron Hick	sct SCO	Brentford	DF	21	9	9	713	0	0	0	5	0	0.2	0.2	0.1	9	21	13	0	0	0
3	Aaron Ran	eng ENG	Arsenal	GK	25	6	6	540	0	0	0	0	0	0	0	0	0	2	0	0	0	0
4	Aaron Ran	eng ENG	Burnley	MF,FW	20	14	5	527	0	0	0	1	0	0.3	0.3	0.4	8	8	17	0	0	0
5	Aaron Wa	eng ENG	Manchest	DF	25	22	20	1780	0	0	2	4	0	0.1	0.1	1.5	30	77	54	0	0.1	0.1
6	Abdoulaye ml	MLI	Everton	FW,MF	30	32	32	2629	7	0	1	7	0	8.8	8.8	2.9	55	97	140	0.24	0.03	0.27
7	Adam Lall	eng ENG	Brighton	MF,FW	35	25	13	850	0	0	1	2	0	0.8	0.8	1.7	9	39	55	0	0.11	0.11
8	Adam Smit	eng ENG	Bournemo	DF	32	28	25	2150	0	0	2	6	0	0.1	0.1	1.3	24	89	28	0	0.08	0.08
9	Adam Wet	eng ENG	Brighton	DF	28	15	13	1144	0	0	0	2	0	0.4	0.4	0.1	16	80	2	0	0	0
10	Adam Wha	eng ENG	Crystal Pal	MF	19	16	15	1297	0	0	3	2	0	0.3	0.3	2.4	14	79	10	0	0.21	0.21
11	Adama Tra	es ESP	Fulham	FW,MF	27	17	1	377	2	0	3	2	0	1.5	1.5	0.7	29	16	48	0.48	0.72	1.19
12	Albert Sam	be BEL	Luton Tow	MF	23	17	16	1303	1	0	3	4	0	0.6	0.6	1.4	29	72	13	0.07	0.21	0.28
13	Alejandro	ar ARG	Manchest	FW	19	36	30	2565	7	0	4	4	0	8.4	8.3	5.1	178	62	281	0.25	0.14	0.39
14	Alex Iwobi	ng NGA	Everton	MF	27	2	2	140	0	0	0	0	0	0.3	0.3	0.2	6	4	8	0	0	0
15	Alex Iwobi	ng NGA	Fulham	FW,MF	27	30	25	2192	5	0	2	2	0	5.3	5.3	4.7	109	147	175	0.21	0.08	0.29
16	Alex Scott	eng ENG	Bournemo	MF	19	23	11	1014	1	0	1	3	0	0.7	0.7	1.7	24	50	35	0.09	0.09	0.18
17	Alexander	se SWE	Newcastle	FW	23	30	27	2255	16	5	2	1	0	20.3	15.6	3.7	68	71	129	0.84	0.08	0.92
18	Alexis Mac	ar ARG	Liverpool	MF	24	33	31	2599	4	1	5	7	1	3.7	2.9	3.6	44	209	48	0.17	0.17	0.35
19	Alfie Doug	eng ENG	Luton Tow	DF	23	37	34	2925	2	0	8	5	0	1.3	1.3	6.3	94	106	164	0.06	0.25	0.31
20	Alfie Gilch	eng ENG	Chelsea	DF	19	11	2	207	1	0	0	1	0	0.3	0.3	0	1	7	4	0.43	0	0.43
21	Alisson	br BRA	Liverpool	GK	30	28	28	2520	0	0	0	1	0	0	0	0	0	5	0	0	0	0
22	Alphonse A	fr FRA	West Ham	GK	30	31	31	2699	0	0	0	1	0	0	0	0	0	1	0	0	0	0
23	Amad Diall	ci CIV	Manchest	FW,MF	21	9	3	390	1	0	1	1	0	0.5	0.5	1.1	11	10	31	0.23	0.23	0.46
24	Amadou O	be BEL	Everton	MF	21	30	23	2091	2	0	0	5	0	2	2	2.7	21	107	39	0.09	0	0.09
25	Amari'i Bel	jm JAM	Luton Tow	DF	29	21	21	1720	0	0	0	0	0	0.3	0.3	0.5	32	76	10	0	0	0

hình 1: file result.csv

2. **Tìm dữ liệu theo các yêu cầu sau :**

a. tìm top ba cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.

```
def load_data(filepath):
    data = pd.read_csv(filepath)
    return data
```

Lấy dữ liệu từ hàm load_data với tham số filepath là đường dẫn đến nơi lưu file results.csv. Trong đó dùng hàm pd.read_csv(filepath) có chức năng đọc dữ liệu từ một file csv.

```
def find_top_3(data, columns):
    for col in columns:
        if pd.api.types.is_numeric_dtype(data[col]):
            print(f"Top 3 highest players for {col}:")
            print(data.nlargest(3, col)[['player', col]])
            print()
            print(f"Top 3 lowest players for {col}:")
            print(data.nsmallest(3, col)[['player', col]])
            print("\n" + "-" * 50 + "\n")
```

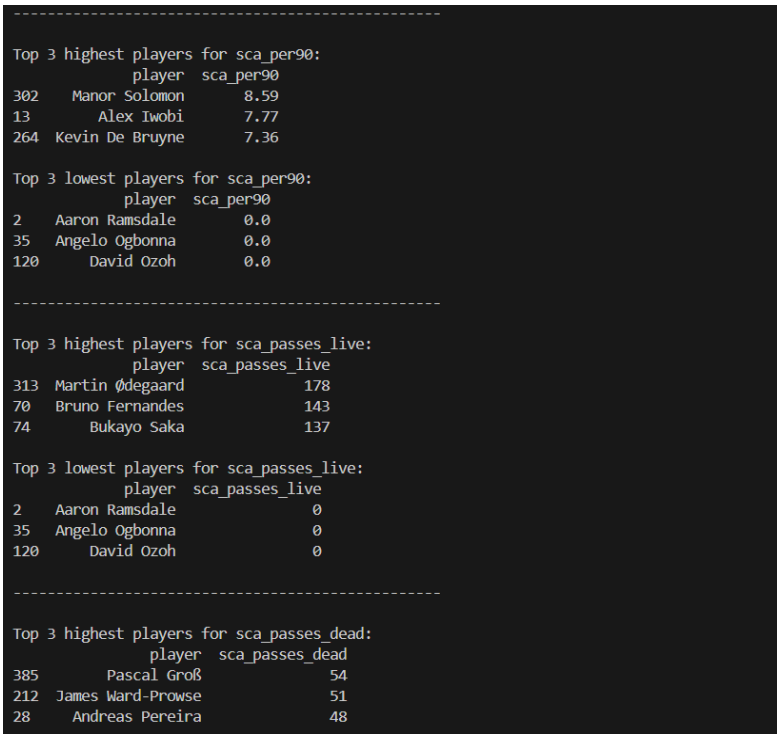
Hàm trên nhận hai tham số là data: tất cả dữ liệu từ file csv và columns: tất cả các cột trừ bốn cột name, team, nation, position không có dữ liệu thuộc kiểu numeric bằng lệnh columns = data.columns[3:]. Tạo ra hai dictionary để lưu lần lượt 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số theo kiểu key = dictionary và value = dataframe với ba hàng dữ liệu.

Duyệt lần lượt qua các cột và kiểm tra xem cột đó có kiểu dữ liệu là số không thông qua hàm của Pandas: pd.api.types.is_numeric_dtype(). Nếu thỏa mãn top ba cầu thủ có điểm cao nhất và thấp nhất được in ra màn hình qua hai hàm sau:

data.nlargest(3, col)[['player', col]] : là hàm của Pandas dùng để tìm ra n hàng có giá trị lớn nhất theo cột columns và trả về dữ liệu dạng một dataframe. Trong trường hợp này, n=3 nghĩa là lấy 3 hàng có giá trị lớn nhất. [['player',

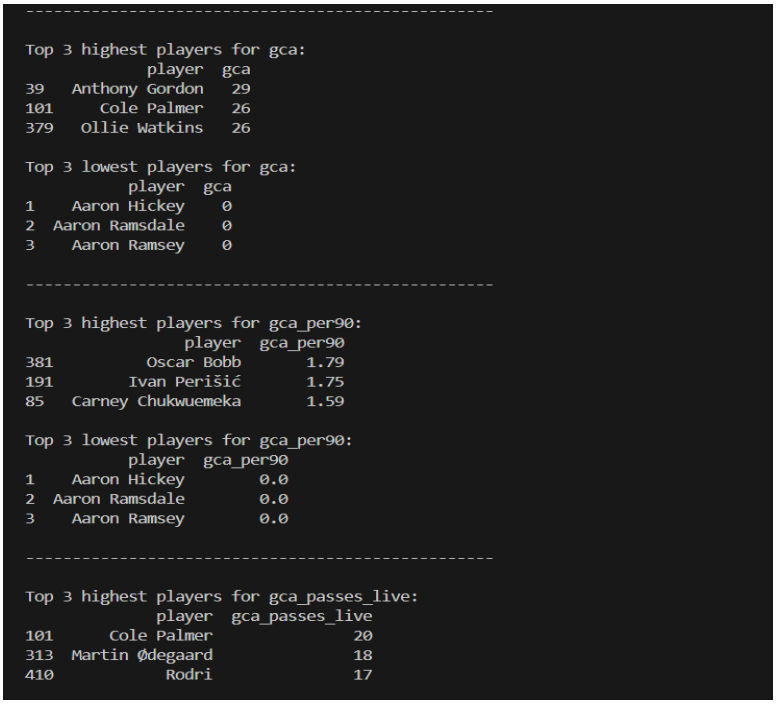
col]] có chức năng lọc lại chỉ lấy hai cột để giữ lại thông tin về tên cầu thủ và chỉ số đang xét làm cho kết quả gọn gang và dễ đọc hơn.

Tương tự với hàm bên trên thì hàm data.nsmallest(3, col)[['player', col]] sẽ lấy top ba cầu thủ được chấm điểm thấp nhất ở chỉ đang xét.



```
--------------------------------------------------

Top 3 highest players for sca_per90:
            player  sca_per90
302    Manor Solomon       8.59
13        Alex Iwobi       7.77
264  Kevin De Bruyne       7.36

Top 3 lowest players for sca_per90:
            player  sca_per90
2    Aaron Ramsdale        0.0
35   Angelo Ogbonna        0.0
120      David Ozoh        0.0

--------------------------------------------------

Top 3 highest players for sca_passes_live:
            player  sca_passes_live
313  Martin Ødegaard              178
70   Bruno Fernandes              143
74       Bukayo Saka              137

Top 3 lowest players for sca_passes_live:
            player  sca_passes_live
2    Aaron Ramsdale                0
35   Angelo Ogbonna                0
120      David Ozoh                0

--------------------------------------------------

Top 3 highest players for sca_passes_dead:
              player  sca_passes_dead
385       Pascal Groß               54
212  James Ward-Prowse               51
28     Andreas Pereira               48
```

hình 2.1: top three highest and lowest



```
--------------------------------------------------

Top 3 highest players for gca:
              player  gca
39    Anthony Gordon   29
101      Cole Palmer   26
379    Ollie Watkins   26

Top 3 lowest players for gca:
            player  gca
1     Aaron Hickey    0
2   Aaron Ramsdale    0
3     Aaron Ramsey    0

--------------------------------------------------

Top 3 highest players for gca_per90:
                player  gca_per90
381         Oscar Bobb       1.79
191       Ivan Perišić       1.75
85   Carney Chukwuemeka       1.59

Top 3 lowest players for gca_per90:
            player  gca_per90
1     Aaron Hickey        0.0
2   Aaron Ramsdale        0.0
3     Aaron Ramsey        0.0

--------------------------------------------------

Top 3 highest players for gca_passes_live:
              player  gca_passes_live
101      Cole Palmer               20
313  Martin Ødegaard               18
410           Rodri                17
```

hình 2.2: top three highest and lowest

b. 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.

Hàm calculate_statistics(data) gồm hai phần: tìm trung vị, trung bình, độ lệch chuẩn của tất cả các cầu thủ và tìm trung vị, trung bình, độ lệch chuẩn cho từng đội bóng.

```
tmp_df = data.copy()
club_list = data['team'].unique().tolist()
numeric_df = data.apply(pd.to_numeric, errors='coerce')
summary_stats = pd.DataFrame()
overall_stats = {}
```

club_list lưu một danh sách tên các câu lạc bộ. numeric_df là một dataframe chứa các dữ liệu đã được chuyển về kiểu số nếu ko phải kiểu số nó sẽ được chuyển thành NaN qua hàm apply của Pandas. Khởi tạo một dataframe rỗng summary_stats để lưu dữ liệu cuối cùng trước khi chuyển sang file results2.csv.

Tạo một từ điển rỗng overall_stats để lưu trữ các thống kê như trung vị (Median), trung bình (Mean), và độ lệch chuẩn (Standard Deviation) của tất cả các cầu thủ trong toàn bộ giải đấu cho từng chỉ số.

```
for col in numeric_df.columns:
    if not numeric_df[col].isnull().all():
        overall_stats[f'{col}_Median'] = numeric_df[col].median()
        overall_stats[f'{col}_Mean'] = numeric_df[col].mean()
        overall_stats[f'{col}_Std'] = numeric_df[col].std()
```

Duyệt lần lượt qua các cột chỉ số và tính trung vị, trung bình và độ lệch chuẩn bằng hàm median(), mean(), std() của thư viện pandas rồi lưu vào overall_stats.

```
overall_stats_df = pd.DataFrame([overall_stats], index=["all"])
summary_stats = pd.concat([summary_stats, overall_stats_df])
```

Tạo mới một dataframe chứa một phần tử với dữ liệu là overall_stats và chỉ định tên cho chỉ mục của dataframe này là ‘all’ để biểu diễn là đây là thống

kê cho tất cả các đội. Dùng hàm concat của thư viện Pandas để nối hai dataframe vào nhau.

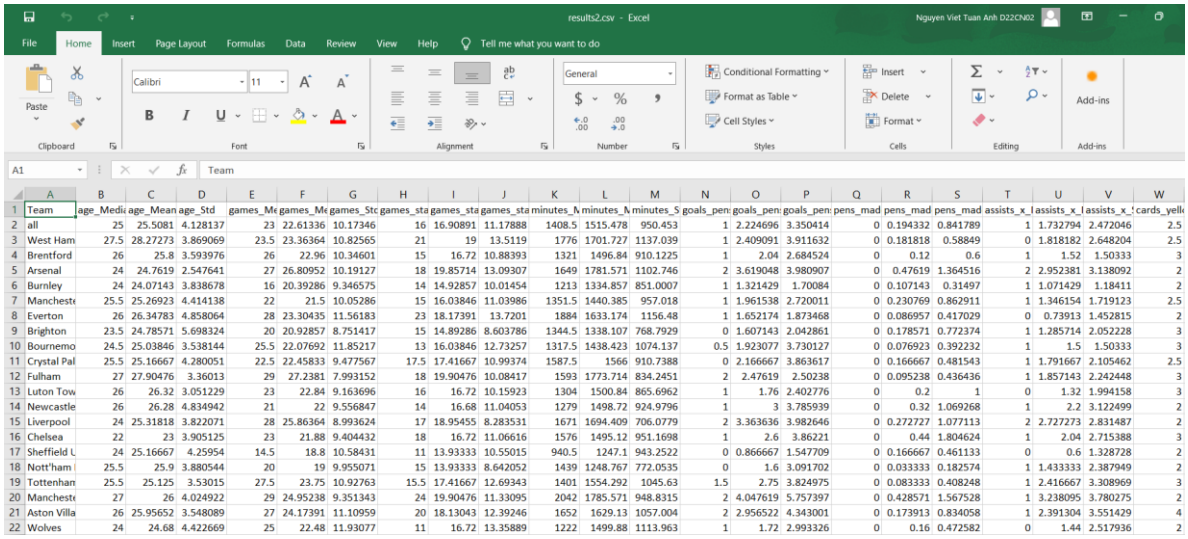
```
col_to_convert = numeric_df.columns.difference(['team', 'STT'])
numeric_df[col_to_convert]=numeric_df[col_to_convert].apply(pd.to_numeric, errors='coerce')
```

Do cần tính toán trung vị , trung bình và độ lệch chuẩn cho từng đội bóng nên hai dòng trên sẽ chịu trách nhiệm tạo ra một dataframe kiểu numeric chứa tất cả dữ liệu và thêm cột team để xác định các đội. col_to_convert sẽ chứa các cột cần chuyển sang kiểu số ngoại trừ hai cột 'team' và 'stt'. Numeric_df sẽ chuyển tất cả các cột ngoại trừ hai cột trên sang kiểu số bằng hàm apply.

Duyệt lần lượt trong danh sách club_list chứa tên các đội bóng và lọc lấy ra dữ liệu của từng đội bóng numeric_df[tmp_df['team'] == club] kiểu dataframe . Tạo mới một biến kiểu từ điển club_stats để lưu từng giá trị trung vị, trung bình và độ lệch chuẩn của từng cột dữ liệu của mỗi đội bóng.

```
for col in club_data.columns:
    if not club_data[col].isnull().all():
        club_stats[f'{col}_Median'] = club_data[col].median()
        club_stats[f'{col}_Mean'] = club_data[col].mean()
        club_stats[f'{col}_Std'] = club_data[col].std()

club_stats_df = pd.DataFrame([club_stats], index=[club])
summary_stats = pd.concat([summary_stats, club_stats_df])
summary_stats.to_csv('C: /results2.csv', index_label="Team")
```

Duyệt qua từng cột để lấy các giá trị yêu cầu và lưu vào club_stats. Tiếp tục chuyển club_stats thành một dataframe với một trường dữ liệu duy nhất và ghép dataframe này vào summary_stats. Chuyển summary_stats thành file csv:



Team	age_Medi	age_Mean	age_Std	games_M	games_M	games_Stc	games_sta	games_sta	games_sta	minutes_N	minutes_N	minutes_S	goals_pen	goals_pen	goals_pen	pens_mad	pens_mad	pens_mad	assists_x_	assists_x_	assists_x_	cards_yell
all	25	25.5081	4.128137	23	22.61336	10.17346	16	16.90891	11.17888	1408.5	1515.478	950.453	1	2.224696	3.350414	0	0.194332	0.841789	1	1.732794	2.472046	2.5
West Ham	27.5	28.27273	3.869069	23.5	23.36364	10.82565	21	19	13.5119	1776	1701.727	1137.039	1	2.409091	3.911632	0	0.181818	0.58849	0	1.818182	2.648204	2.5
Brentford	26	25.8	3.593976	26	22.96	10.34601	15	16.72	10.88393	1321	1496.84	910.1225	1	2.04	2.684524	0	0.12	0.6	1	1.52	1.50333	3
Arsenal	24	24.7619	2.547641	27	26.80952	10.19127	18	19.85714	13.09307	1649	1781.571	1102.746	2	3.619048	3.980907	0	0.47619	1.364516	2	2.952381	3.138092	2
Burnley	24	24.07143	3.838678	16	20.39286	9.346575	14	14.92857	10.01454	1213	1334.857	851.0007	1	1.321429	1.70084	0	0.107143	0.31497	1	1.071429	1.18411	2
Mancheste	25.5	25.26923	4.414138	22	21.5	10.05286	15	16.03846	11.03986	1351.5	1440.385	957.018	1	1.961538	2.720011	0	0.230769	0.862911	1	1.346154	1.719123	2.5
Everton	26	26.34783	4.858064	28	23.30435	11.56183	23	18.17391	13.7201	1884	1633.174	1156.48	1	1.652174	1.873468	0	0.086957	0.417029	0	0.73913	1.452815	2
Brighton	23.5	24.78571	5.698324	20	20.92857	8.751417	15	14.89286	8.603786	1344.5	1338.107	768.7929	0	1.607143	2.042861	0	0.178571	0.772374	1	1.285714	2.052228	3
Bournemo	24.5	25.03846	3.538144	25.5	22.07692	11.85217	13	16.03846	12.73257	1317.5	1438.423	1074.137	0.5	1.923077	3.730127	0	0.076923	0.392232	1	1.5	1.50333	3
Crystal Pal	25.5	25.16667	4.280051	22.5	22.45833	9.477567	17.5	17.41667	10.99374	1587.5	1566	910.7388	0	2.166667	3.863617	0	0.166667	0.481543	1	1.791667	2.105462	2.5
Fulham	27	27.90476	3.36013	29	27.2381	7.993152	18	19.90476	10.08417	1593	1773.714	834.2451	2	2.47619	2.50238	0	0.095238	0.436436	1	1.857143	2.242448	3
Luton Tow	26	26.32	3.051229	23	22.84	9.163696	16	16.72	10.15923	1304	1500.84	865.6962	1	1.76	2.402776	0	0.2	1	0	1.32	1.994158	3
Newcastle	26	26.28	4.834942	21	22	9.556847	14	16.68	11.04053	1279	1498.72	924.9796	1	3	3.785939	0	0.32	1.069268	1	2.2	3.122499	2
Liverpool	24	25.31818	3.822071	28	25.86364	8.993624	17	18.95455	8.283531	1671	1694.409	706.0779	2	3.363636	3.982646	0	0.272727	1.077113	2	2.727273	2.831487	2
Chelsea	22	23	3.905125	23	21.88	9.404432	18	16.72	11.06616	1576	1495.12	951.1698	1	2.6	3.86221	0	0.44	1.804624	1	2.04	2.715388	3
Sheffield U	24	25.16667	4.25954	14.5	18.8	10.58431	11	13.93333	10.55015	940.5	1247.1	943.2522	0	0.866667	1.547709	0	0.166667	0.461133	0	0.6	1.328728	2
Nott'ham	25.5	25.9	3.880544	20	19	9.955071	15	13.93333	8.642052	1439	1248.767	772.0535	0	1.6	3.091702	0	0.033333	0.182574	1	1.433333	2.387949	2
Tottenhan	25.5	25.125	3.53015	27.5	23.75	10.92763	15.5	17.41667	12.69343	1401	1554.292	1045.63	1.5	2.75	3.824975	0	0.083333	0.408248	1	2.416667	3.308969	3
Mancheste	27	26	4.024922	29	24.95238	9.351343	24	19.90476	11.33095	2042	1785.571	948.8315	2	4.047619	5.757397	0	0.428571	1.567528	1	3.238095	3.780275	2
Aston Villa	26	25.95652	3.548089	27	24.17391	11.10959	20	18.13043	12.39246	1652	1629.13	1057.004	2	2.956522	4.343001	0	0.173913	0.834058	1	2.391304	3.551429	4
Wolves	24	24.68	4.422669	25	22.48	11.93077	11	16.72	13.35889	1222	1499.88	1113.963	1	1.72	2.993326	0	0.16	0.472582	0	1.44	2.517936	2

hình 2.3: file results2.csv

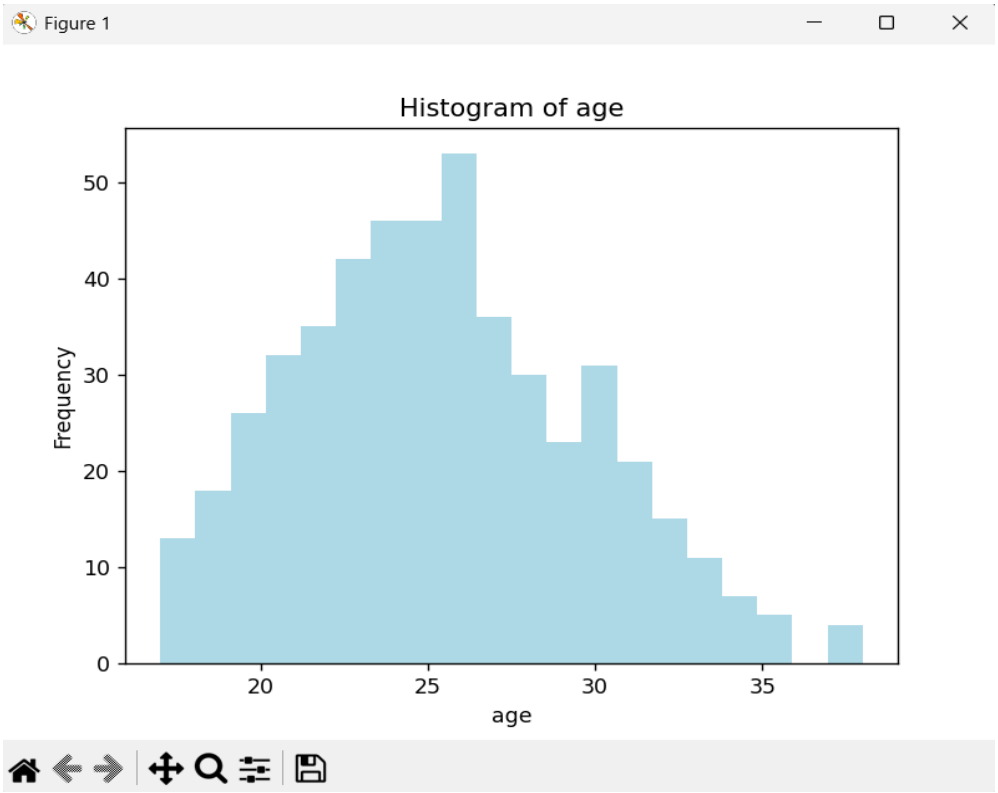
c. Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.

Hàm plot_histograms nhận vào hai tham số là một dataframe và một danh sách các cột cần vẽ histogram. Columns ở đây đã được loại bỏ cột name, team, nation, position.

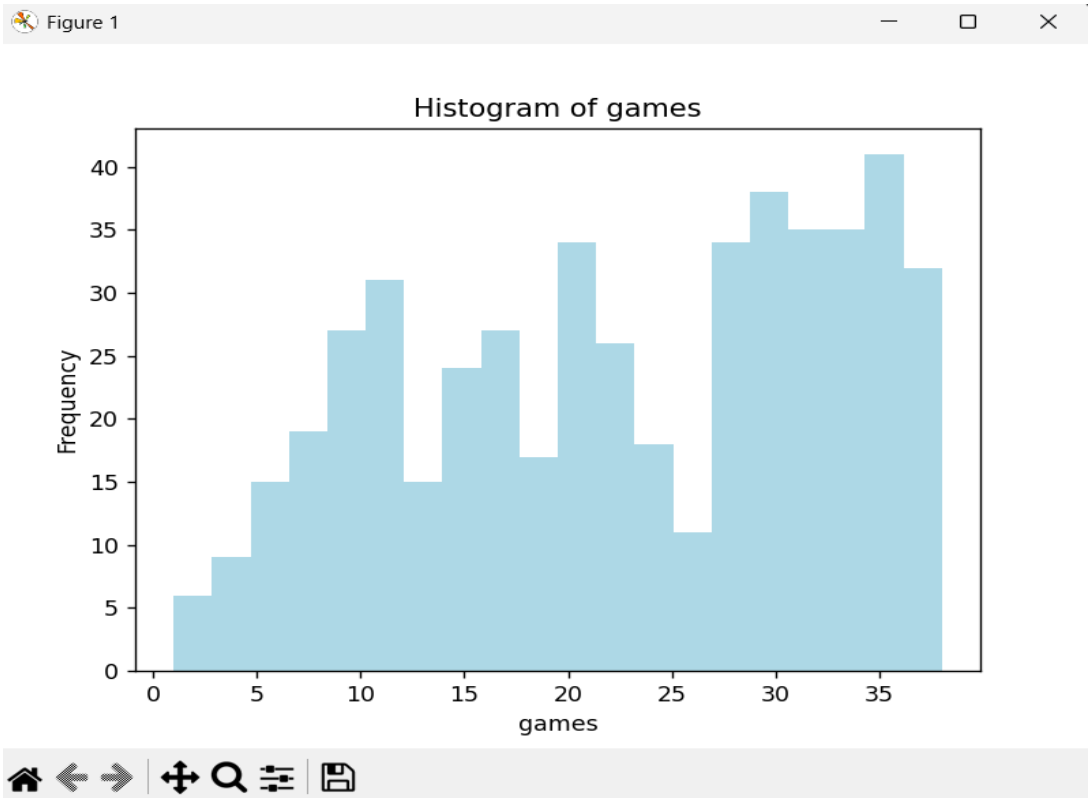
```
def plot_histograms(data, columns):
    for col in columns:
        if pd.api.types.is_numeric_dtype(data[col]):
            plt.figure()
            plt.hist(data[col].dropna(), bins=20, color='lightblue')
            plt.title(f'Histogram of {col}')
            plt.xlabel(col)
            plt.ylabel('Frequency')
            plt.show()
```

Trong hàm plot_histogram duyệt qua tất cả các cột bỏ qua các cột không phải kiểu số và tạo mới một biểu đồ mới cho mỗi cột bằng plt.figure() của thư viện Matplotlib. Cài đặt các thông số cho biểu đồ với 20 khoảng dữ liệu, màu xanh nhạt và đồng thời loại bỏ các giá trị rỗng. Đặt nhãn cho trục x và trục y lần lượt là Frequency và tên cột của cột đang xét.

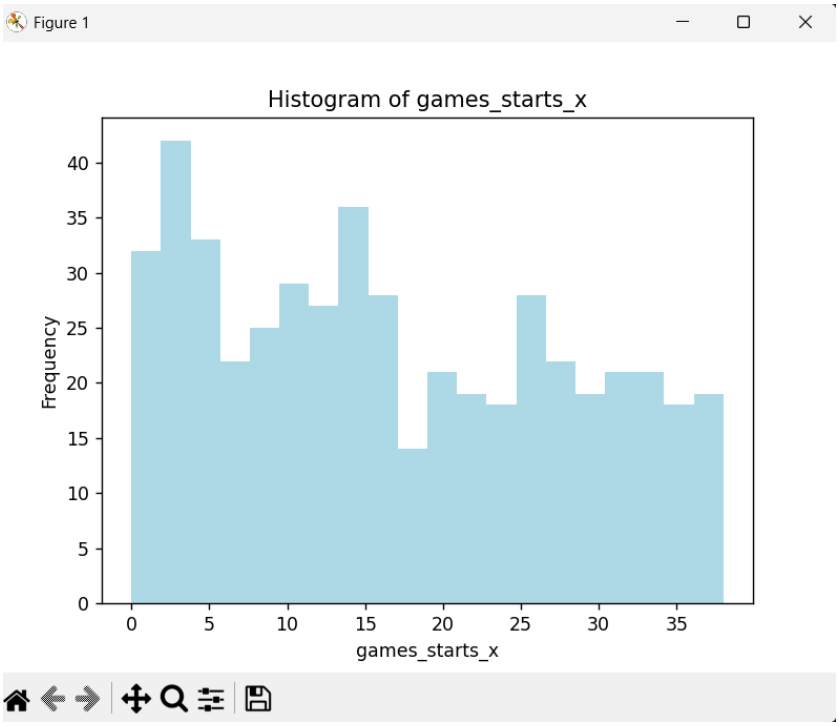
Một số biểu đồ trong tất cả dữ liệu:



hình 2.4: histogram of age



hình 2.5: histogram of games



hình 2.6: histogram of games starts

d. 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.

```
def calculate_team_performance(data):
    numeric_data = data.select_dtypes(include='number')
    numeric_data['Team'] = data['team']
    team_performance = numeric_data.groupby('Team').mean()
    overall_mean = team_performance.mean(axis=1)
    best_team = overall_mean.idxmax()
    return best_team
```

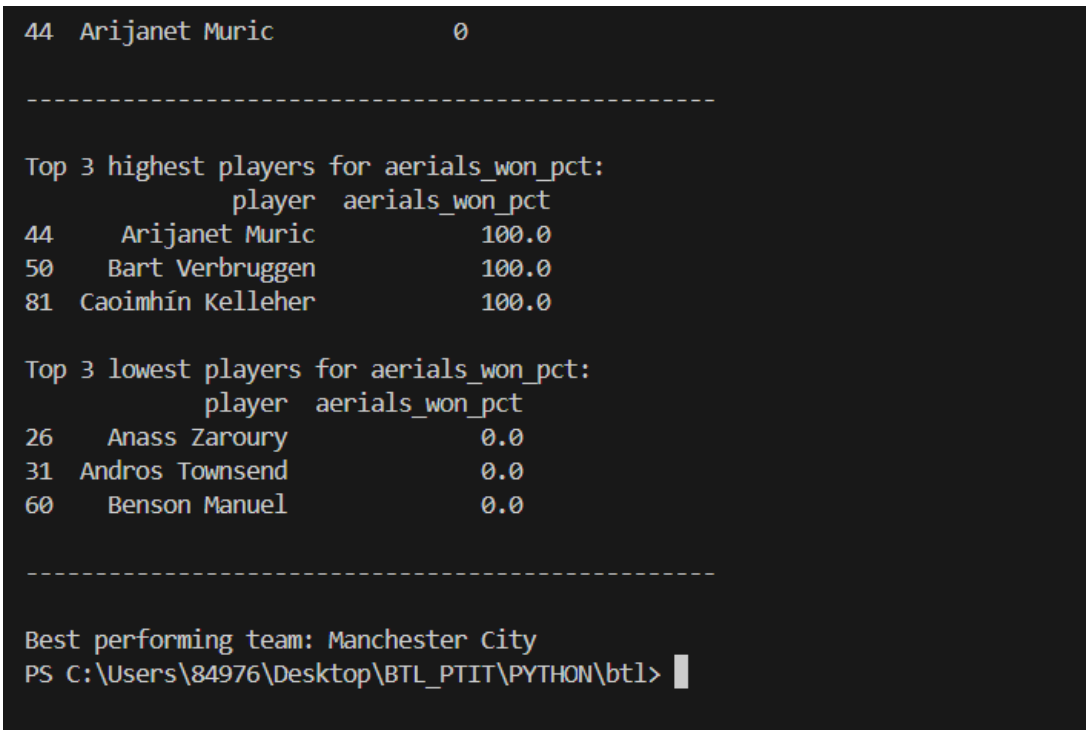
Hàm calculate_team_performance nhận vào một tham số là một dataframe chứa toàn bộ dữ liệu. Numeric_data là một dataframe chỉ chứa các cột dữ liệu số từ tham số đầu vào và được thêm cột ‘team’. Điều này cho phép ghép nhóm dữ liệu theo đội. numeric_data.groupby('Team').mean() sẽ tạo ra một dataframe mới với chỉ số là tên các đội và giá trị trung bình của các chỉ số tương ứng.

```
overall_mean = team_performance.mean(axis=1)
```

```
best_team = overall_mean.idxmax()
```

Ở bước này, hàm tính trung bình của các trung bình của từng đội (theo hàng) để có được một giá trị trung bình tổng thể cho mỗi đội. Kết quả là một Series overall_mean, trong đó mỗi giá trị tương ứng với một đội. Đội có phong độ tốt nhất sẽ được xác định bằng hàm idxmax() hay đội có giá trị trung bình lớn nhất trong overall_mean.

Đội có phong độ cao nhất màu giải 2023-2024 giải bóng đá ngoại hạng anh là câu lạc bộ Manchester City.



hình 2.7: best performing team

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

```
def load_data(filepath):
    data = pd.read_csv(filepath)
    return data
data  = load_data('C:/Users/results.csv')
```

Hàm load_data nhận vào một đường dẫn đến tệp csv và tải dữ liệu vào một dataframe ở đây là data. Sau đó chuẩn hóa dữ liệu trước khi phân cụm:

```
numeric_data = data.select_dtypes(include='number')
imputer = SimpleImputer(strategy='mean')
numeric_data = imputer.fit_transform(numeric_data)
```

Tạo ra một dataframe numeric_data để lưu dữ liệu sua khi đã lọc ra các cột có kiểu dữ liệu số và đối tượng imputer từ thư viện scikit-learn để thay thế các giá trị NaN bằng giá trị trung bình của từng cột tránh gây lỗi khi phân cụm. Fit_transform() là hàm dùng để áp dụng các phương pháp thay thế này vào numeric_data.

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_data)
```

Khởi tạo đối tượng StandardScaler từ thư viện scikit-learn sẽ chuẩn hóa dữ liệu đưa tất cả các biết cần so sánh về một thang đo và lưu vào scaled_data.

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

Đoạn mã trên sử dụng phương pháp Elbow để xác định số cụm tối ưu cho thuật toán K-means. WCSS(within cluster sum of squares) là tổng khoảng cách bình phương của mỗi điểm đến tâm cụm tương ứng. Danh sách wcss lưu cá trị WCSS cho từng số lượng cụm k từ 1 đến 10:

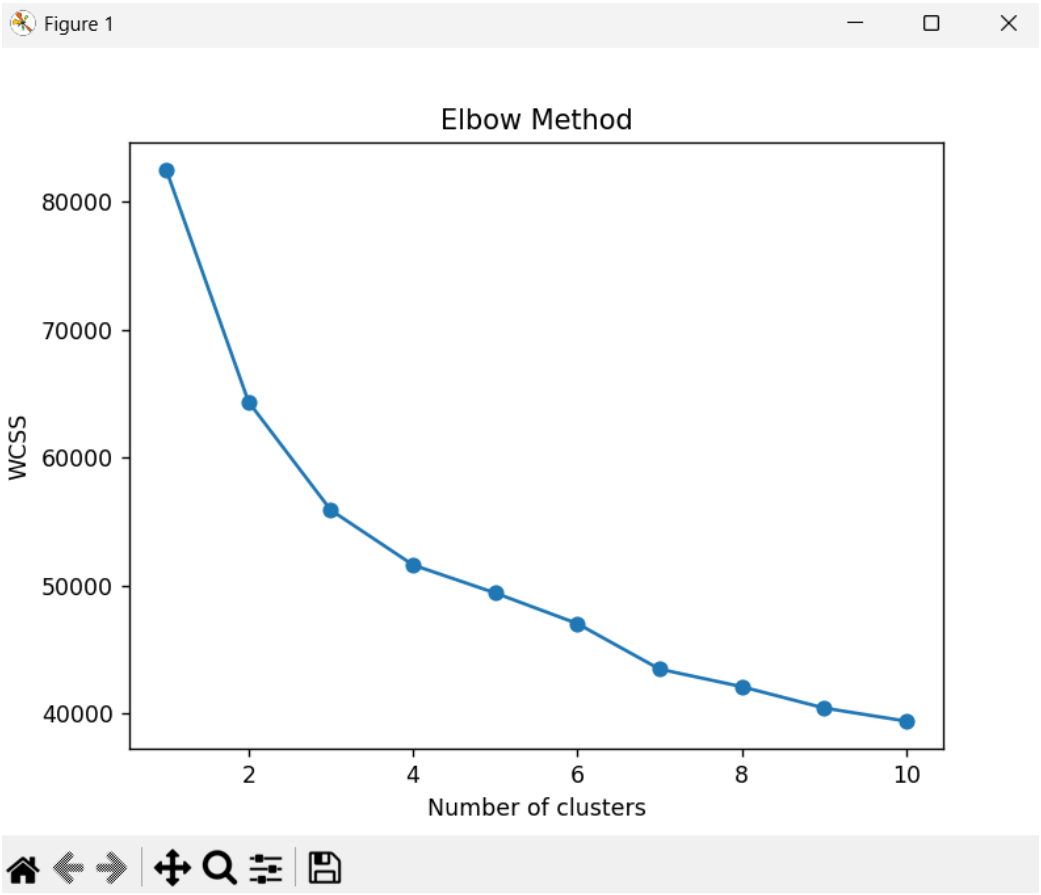
KMeans(n_clusters=k, random_state=42) tạo một mô hình KMeans với k cụm và thiết lập random_state=42 để kết quả phân cụm là nhất quán. kmeans.fit(scaled_data): Huấn luyện mô hình KMeans trên dữ liệu đã được chuẩn hóa (scaled_data). Trong quá trình này, thuật toán K-means sẽ tìm kiếm k tâm cụm và phân chia các điểm dữ liệu vào các cụm tương ứng.

kmeans.inertia_: Đây là giá trị WCSS cho số lượng cụm k hiện tại. inertia_ trả về tổng các khoảng cách bình phương từ mỗi điểm dữ liệu đến tâm cụm của nó. Lưu giá trị này vào danh sách wcss để vẽ biểu đồ Elbow:

```
plt.plot(range(1, 11), wcss, marker='o')
```

```
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Hàm plot nhận vào bat ham số lần lượt: trục x biểu diễn số lượng cụm từ một đến 10, trục y là danh sách giá trị WCSS đã tính ở trên, marker là kí hiệu để biểu thị từng điểm dữ liệu trên biểu đồ.



hình 3.1: Elbow Method

Dựa trên biểu đồ Elbow thì điểm gấp khúc rõ rệt nhất trong biểu đồ này là lúc số lượng cụm là 3 vì từ số cụm 3 trở đi thì mức giảm của WCSS không đáng kể, nghĩa là việc thêm cụm sẽ không làm giảm nhiều độ phân tán trong mỗi cụm. Số cụm tôi ưu ở đây sẽ là 3.

```
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
pca = PCA(n_components=2)
```

```
pca_data = pca.fit_transform(scaled_data)
```

Áp dụng thuật toán K-means để phân cụm dữ liệu và sử dụng PCA để giảm chiều dữ liệu xuống 2D nhằm dễ dàng trực quan hóa. optimal_k là số lượng cụm tối ưu đã được chọn nhờ phương pháp Elbow.

KMeans(n_clusters=optimal_k, random_state=42) tạo mô hình K-means với ba cụm và thiết lập random_state=42 để đảm bảo tính nhất quán trong các lần chạy. fit_predict(scaled_data): Huấn luyện mô hình K-means trên dữ liệu đã được chuẩn hóa (scaled_data) và trả về nhãn cụm cho từng điểm dữ liệu. Lưu nhãn cụm vào cột 'Cluster' trong dataframe data để có thể phân tích và trực quan hóa.

Khởi tạo một đối tượng PCA với n_component = 2 nghĩa là giảm chiều dữ liệu xuống 2 thành phần. pca_data sẽ lưu dữ liệu mới sau khi đã giảm chiều dữ liệu thông qua hàm fit_transform áp dụng lên dữ liệu đã chuẩn hóa.

➔ Sau đoạn mã trên thì dữ liệu đã được phân vào 3 cụm, và chiều của dữ liệu đã giảm xuống còn 2 chiều để dễ dàng vẽ biểu đồ và quan sát sự phân cụm.

```
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=data['Cluster'], cmap='viridis')

plt.title('K-means Clustering with PCA')

plt.xlabel('PCA 1')

plt.ylabel('PCA 2')

plt.show()
```

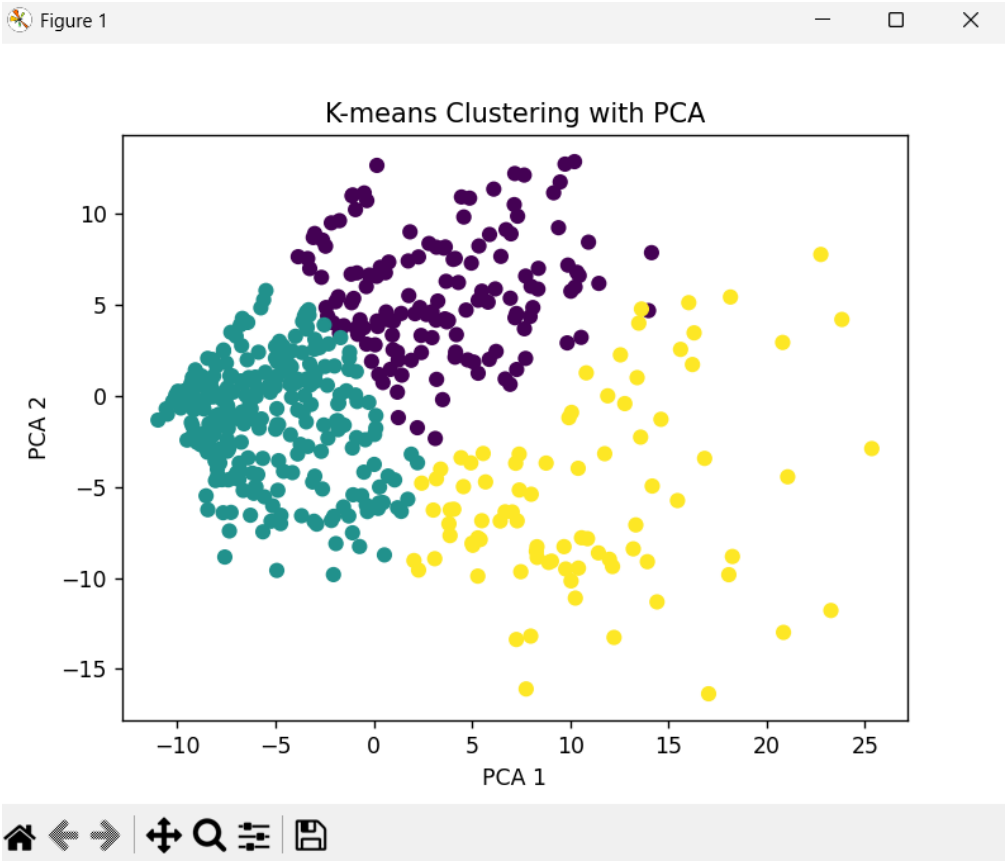
Hàm scatter từ thư viện Matplotlib cho ra một biểu đồ phân tán với bốn tham số lần lượt là:

pca_data[:, 0]: Trục x của biểu đồ là thành phần chính thứ nhất (PCA 1).

pca_data[:, 1]: Trục y của biểu đồ là thành phần chính thứ hai (PCA 2).

c=data['Cluster']: Màu sắc của các điểm dữ liệu sẽ được đặt dựa trên nhãn cụm của chúng (cột Cluster trong data).

cmap='viridis': Sử dụng bảng màu "viridis" để mã hóa các cụm khác nhau bằng màu sắc.



Hình 3.2: K-means Clustering with PCA

III. Tài liệu tham khảo.

1. https://scikitlearn.org/1.5/modules/generated/sklearn.cluster.KMeans.
2. https://codelearn.io/sharing/xu-ly-du-lieu-voi-pandas-trong-python
3. https://www.tutorialspoint.com/beautiful_soup/index.htm
4. https://viblo.asia/p/cao-du-lieu-doanh-nghiep-voi-beautiful-soup-mot-cach-cuc-ky-don-gian-V3m5WjgblO7
5. htmlhttps://pandas.pydata.org/docs/