Bài 1:

Đầu tiên tạo một list chứa url của 20 đội tuyển.

```
urls = [
    "https://fbref.com/en/squads/b8fd03ef/2023-2024/Manchester-City-Stats",
    "https://fbref.com/en/squads/18bb7c10/2023-2024/Arsenal-Stats",
    "https://fbref.com/en/squads/822bd0ba/2023-2024/Liverpool-Stats",
    "https://fbref.com/en/squads/8602292d/2023-2024/Aston-Villa-Stats",
    "https://fbref.com/en/squads/361ca564/2023-2024/Tottenham-Hotspur-Stats",
    "https://fbref.com/en/squads/cff3d9bb/2023-2024/Chelsea-Stats",
    "https://fbref.com/en/squads/b2b47a98/2023-2024/Newcastle-United-Stats",
    "https://fbref.com/en/squads/19538871/2023-2024/Manchester-United-Stats",
    "https://fbref.com/en/squads/7c21e445/2023-2024/West-Ham-United-Stats",
    "https://fbref.com/en/squads/47c64c55/2023-2024/Crystal-Palace-Stats",
    "https://fbref.com/en/squads/d07537b9/2023-2024/Brighton-and-Hove-Albion-Stats",
    "https://fbref.com/en/squads/4ba7cbea/2023-2024/Bournemouth-Stats",
    "https://fbref.com/en/squads/fd962109/2023-2024/Fulham-Stats",
    "https://fbref.com/en/squads/8cec06e1/2023-2024/Wolverhampton-Wanderers-Stats",
    "https://fbref.com/en/squads/d3fd31cc/2023-2024/Everton-Stats",
    "https://fbref.com/en/squads/cd051869/2023-2024/Brentford-Stats",
    "https://fbref.com/en/squads/e4a775cb/2023-2024/Nottingham-Forest-Stats",
    "https://fbref.com/en/squads/e297cd13/2023-2024/Luton-Town-Stats",
    "https://fbref.com/en/squads/943e8050/2023-2024/Burnley-Stats",
    "https://fbref.com/en/squads/1df6b87e/2023-2024/Sheffield-United-Stats"
]
```

Duyệt từng url của các đội tuyển, tạo ra 10 list là 10 bảng để chứa dữ liệu cần lấy, duyệt từng bảng để lấy các dữ liệu cần lấy, sau đó gộp các bảng dữ liệu với nhau thông qua tên cầu thủ.

Trong bảng Standard có hàm if để những cầu thủ thi đấu dưới 90 phút thì dữ liệu phút sẽ là N/A

Tạo một list dataframe để có thể lưu dữ liệu các bảng đã được gộp để lấy dữ liệu cầu thủ của từng đội

```python
dataframes = []
for url in urls:
    r = requests.get(url)
    soup = bs(r.content, features: 'html.parser')

    team_name = url.split("/")[-1].replace("-Stats", "")

    standard_data = []
    goalkeeping_data = []
    shooting_data = []
    passing_data = []
    pass_types_data = []
    gsc_data = []
    defensive_data = []
    possession_data = []
    playing_time_data = []
    miscellaneous_data = []

    standard_table = soup.find( name: "table", attrs: {"id": "stats_standard_9"})
    standard_rows = standard_table.find_all('tr')[2:]  # Bỏ qua hàng tiêu đề
    for row in standard_rows[:len(standard_rows) - 2]:
        player_th = row.find('th', {'data-stat': 'player'})
        cols = row.find_all('td')
        min_text = cols[5].text.strip().replace(",", "")

        if not min_text.isdigit() or int(min_text) <= 90:
            continue
```

Đây là hàm lấy dữ liệu và gộp các bảng lại với nhau, nếu chỉ số Min là N/A thì xóa hàng đó đi để bỏ qua cầu thủ thi đấu dưới 90 phút

```
df_standard = pd.DataFrame(standard_data)
df_goalkeeping = pd.DataFrame(goalkeeping_data)
df_shooting = pd.DataFrame(shooting_data)
df_passing = pd.DataFrame(passing_data)
df_pass_types = pd.DataFrame(pass_types_data)
df_gsc = pd.DataFrame(gsc_data)
df_defensive = pd.DataFrame(defensive_data)
df_possession = pd.DataFrame(possession_data)
df_playing_time = pd.DataFrame(playing_time_data)
df_miscellaneous = pd.DataFrame(miscellaneous_data)

merged_df = pd.merge(df_standard, df_goalkeeping, on='Player', how='outer', suffixes=('', '_Keeper'))
merged_df = pd.merge(merged_df, df_shooting, on='Player', how='outer', suffixes=('', '_Shooting'))
merged_df = pd.merge(merged_df, df_passing, on='Player', how='outer', suffixes=('', '_Passing'))
merged_df = pd.merge(merged_df, df_pass_types, on='Player', how='outer', suffixes=('', '_PassTypes'))
merged_df = pd.merge(merged_df, df_gsc, on='Player', how='outer', suffixes=('', '_GSC'))
merged_df = pd.merge(merged_df, df_defensive, on='Player', how='outer', suffixes=('', '_Defensive'))
merged_df = pd.merge(merged_df, df_possession, on='Player', how='outer', suffixes=('', '_Possession'))
merged_df = pd.merge(merged_df, df_playing_time, on='Player', how='outer', suffixes=('', '_PlayingTime'))
merged_df = pd.merge(merged_df, df_miscellaneous, on='Player', how='outer', suffixes=('', '_Miscellaneous'))

merged_df.replace( to_replace: '',  value: 'N/A', inplace=True)
merged_df.fillna( value: 'N/A', inplace=True)
for index, row in merged_df.iterrows():
    if row['Min'] == 'N/A':
        merged_df.drop(index, inplace=True)
dataframes.append(merged_df)
time.sleep(5)
```

Đây là hàm ghép các đội tuyển lại thành một bảng dữ liệu duy nhất, chuyển tuổi thành số, sắp xếp theo tên và tuổi, sau đó dán vào file result.csv

Bài 2:

Code tìm cầu thủ top 3 điểm cao nhất, điểm thấp nhất của từng chỉ số rồi in ra

```python
df = pd.read_csv('result.csv')
df['Min'] = df['Min'].str.replace(',', '').astype(int)

numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

top_bottom_players = {}

for col in numeric_columns:
    top_3 = df.nlargest( n: 3, col)[['Player', col]]
    bottom_3 = df.nsmallest( n: 3, col)[['Player', col]]
    top_bottom_players[col] = {
        'Top 3': top_3.to_dict(orient='records'),
        'Bottom 3': bottom_3.to_dict(orient='records')
    }

for stat, players in top_bottom_players.items():
    print(f"\n{stat} - Top 3:")
    for player in players['Top 3']:
        print(f"{player['Player']}: {player[stat]}")

    print(f"\n{stat} - Bottom 3:")
    for player in players['Bottom 3']:
        print(f"{player['Player']}: {player[stat]}")
```

Code tìm trung bình và độ lệch chuẩn của mỗi chỉ số sau đó dán vào file result2.csv

```python
team_stats = df[numeric_columns].groupby(df['Team']).agg(['median', 'mean', 'std'])
team_stats.columns = ['_'.join(col).strip() for col in team_stats.columns.values]
team_stats.to_csv('results2.csv')
```

Code tìm đội bóng cao nhất của mỗi chỉ số, và đội bóng có phong độ tốt nhất

```python
highest_teams = {}
for stat in team_stats.columns:
    highest_teams[stat] = team_stats[stat].idxmax()

for stat, team in highest_teams.items():
    print(f"Đội bóng có {stat.replace('_', ' ')} cao nhất là: {team}")

mean_columns = [col for col in team_stats.columns if 'mean' in col]
best_team = team_stats[mean_columns].mean(axis=1).idxmax()
print(f"\nĐội bóng có phong độ tốt nhất giải Ngoại hạng Anh mùa 2023-2024 là: {best_team}")
```

Code Histogram Plot

```python
numeric_data = df.select_dtypes(include=['float64', 'int64'])

total_histograms = len(numeric_data.columns)

num_figures = 90

hist_per_fig = max(1, total_histograms // num_figures + 1)

for fig_index in range(num_figures):
    fig, axes = plt.subplots(min(hist_per_fig, total_histograms - fig_index * hist_per_fig), ncols: 1,
                             figsize=(12, 6 * min(hist_per_fig, total_histograms - fig_index * hist_per_fig)))

    for i in range(min(hist_per_fig, total_histograms - fig_index * hist_per_fig)):
        col = numeric_data.columns[fig_index * hist_per_fig + i]
        axes[i].hist(numeric_data[col].dropna(), bins=20, color='skyblue', edgecolor='black')
        axes[i].set_title(f'Histogram of {col}', fontsize=16)
        axes[i].set_xlabel(col, fontsize=14)
        axes[i].set_ylabel('Frequency', fontsize=14)
        axes[i].grid(axis='y', alpha=0.75)
    if len(axes) < hist_per_fig:
        for j in range(len(axes), hist_per_fig):
            axes[j].axis('off')

    plt.subplots_adjust(hspace=0.5)
    plt.tight_layout(pad=2.0)
    plt.show()
```

Em chia thành 90 figure để biểu đồ sẽ thoáng hơn.

Vì kết quả của code quá dài nên em chỉ cắt một vài chỉ số của Top đầu và cuối.

```
Age - Top 3:
Ashley Young: 38
Thiago Silva: 38
Łukasz Fabiański: 38

Age - Bottom 3:
Leon Chiwome: 17
Lewis Miley: 17
David Ozoh: 18

MP - Top 3:
André Onana: 38
Bernd Leno: 38
Carlton Morris: 38

MP - Bottom 3:
Alex Iwobi: 2
Ionuț Radu: 2
Matheus Nunes: 2

Starts - Top 3:
André Onana: 38
Bernd Leno: 38
Guglielmo Vicario: 38

Starts - Bottom 3:
David Ozoh: 0
Ivan Perišić: 0
Jesurun Rak Sakyi: 0

Min - Top 3:
André Onana: 3420
Bernd Leno: 3420
Guglielmo Vicario: 3420
```

Đây là một vài của một vài đội bóng và đôi bóng có thành tích cao nhất

```
Đội bóng có Age median cao nhất là: West-Ham-United
Đội bóng có Age mean cao nhất là: West-Ham-United
Đội bóng có Age std cao nhất là: Brighton-and-Hove-Albion
Đội bóng có MP median cao nhất là: Fulham
Đội bóng có MP mean cao nhất là: Fulham
Đội bóng có MP std cao nhất là: Wolverhampton-Wanderers
Đội bóng có Starts median cao nhất là: Manchester-City
Đội bóng có Starts mean cao nhất là: Fulham
Đội bóng có Starts std cao nhất là: Everton
Đội bóng có Min median cao nhất là: Manchester-City
Đội bóng có Min mean cao nhất là: Manchester-City
Đội bóng có Min std cao nhất là: Everton
Đội bóng có 90s median cao nhất là: Manchester-City
Đội bóng có 90s mean cao nhất là: Manchester-City
Đội bóng có 90s std cao nhất là: Everton
Đội bóng có Gls median cao nhất là: Arsenal
Đội bóng có Gls mean cao nhất là: Manchester-City
Đội bóng có Gls std cao nhất là: Manchester-City
Đội bóng có Ast median cao nhất là: Arsenal
Đội bóng có Ast mean cao nhất là: Manchester-City
Đội bóng có Ast std cao nhất là: Manchester-City
Đội bóng có G+A median cao nhất là: Fulham
Đội bóng có G+A mean cao nhất là: Manchester-City
Đội bóng có G+A std cao nhất là: Manchester-City
Đội bóng có G-PK median cao nhất là: Arsenal
Đội bóng có G-PK mean cao nhất là: Manchester-City
Đội bóng có G-PK std cao nhất là: Manchester-City
Đội bóng có PK median cao nhất là: Arsenal
Đội bóng có PK mean cao nhất là: Arsenal
```

```
Đội bóng có onGA median cao nhất là: West-Ham-United
Đội bóng có onGA mean cao nhất là: Sheffield-United
Đội bóng có onGA std cao nhất là: Sheffield-United
Đội bóng có onxG median cao nhất là: Liverpool
Đội bóng có onxG mean cao nhất là: Liverpool
Đội bóng có onxG std cao nhất là: Arsenal
Đội bóng có onxGA median cao nhất là: West-Ham-United
Đội bóng có onxGA mean cao nhất là: West-Ham-United
Đội bóng có onxGA std cao nhất là: West-Ham-United
Đội bóng có Fls median cao nhất là: Everton
Đội bóng có Fls mean cao nhất là: Liverpool
Đội bóng có Fls std cao nhất là: Chelsea
Đội bóng có Fld Miscellaneous median cao nhất là: Manchester-City
Đội bóng có Fld Miscellaneous mean cao nhất là: Tottenham-Hotspur
Đội bóng có Fld Miscellaneous std cao nhất là: Newcastle-United
Đội bóng có Off median cao nhất là: Brentford
Đội bóng có Off mean cao nhất là: Liverpool
Đội bóng có Off std cao nhất là: Liverpool
Đội bóng có Crs Miscellaneous median cao nhất là: Fulham
Đội bóng có Crs Miscellaneous mean cao nhất là: Liverpool
Đội bóng có Crs Miscellaneous std cao nhất là: Luton-Town
Đội bóng có OG median cao nhất là: Arsenal
Đội bóng có OG mean cao nhất là: Sheffield-United
Đội bóng có OG std cao nhất là: Fulham
Đội bóng có Recov median cao nhất là: Liverpool
Đội bóng có Recov mean cao nhất là: Liverpool
Đội bóng có Recov std cao nhất là: Everton
Đội bóng có Aerial Won median cao nhất là: West-Ham-United
Đội bóng có Aerial Won mean cao nhất là: Everton
Đội bóng có Aerial Won std cao nhất là: Everton
Đội bóng có Aerial Lost median cao nhất là: Brentford
Đội bóng có Aerial Lost mean cao nhất là: Bournemouth
Đội bóng có Aerial Lost std cao nhất là: Luton-Town
Đội bóng có Aerial Won% median cao nhất là: Nottingham-Forest
Đội bóng có Aerial Won% mean cao nhất là: Nottingham-Forest
Đội bóng có Aerial Won% std cao nhất là: Manchester-City

Đội bóng có phong độ tốt nhất giải Ngoại hạng Anh mùa 2023-2024 là: Manchester-City
```

Đây là kết quả của một vài Histogram Plot, nó sẽ hiện từng figure khi x đi, em đã tạo 90 figure.

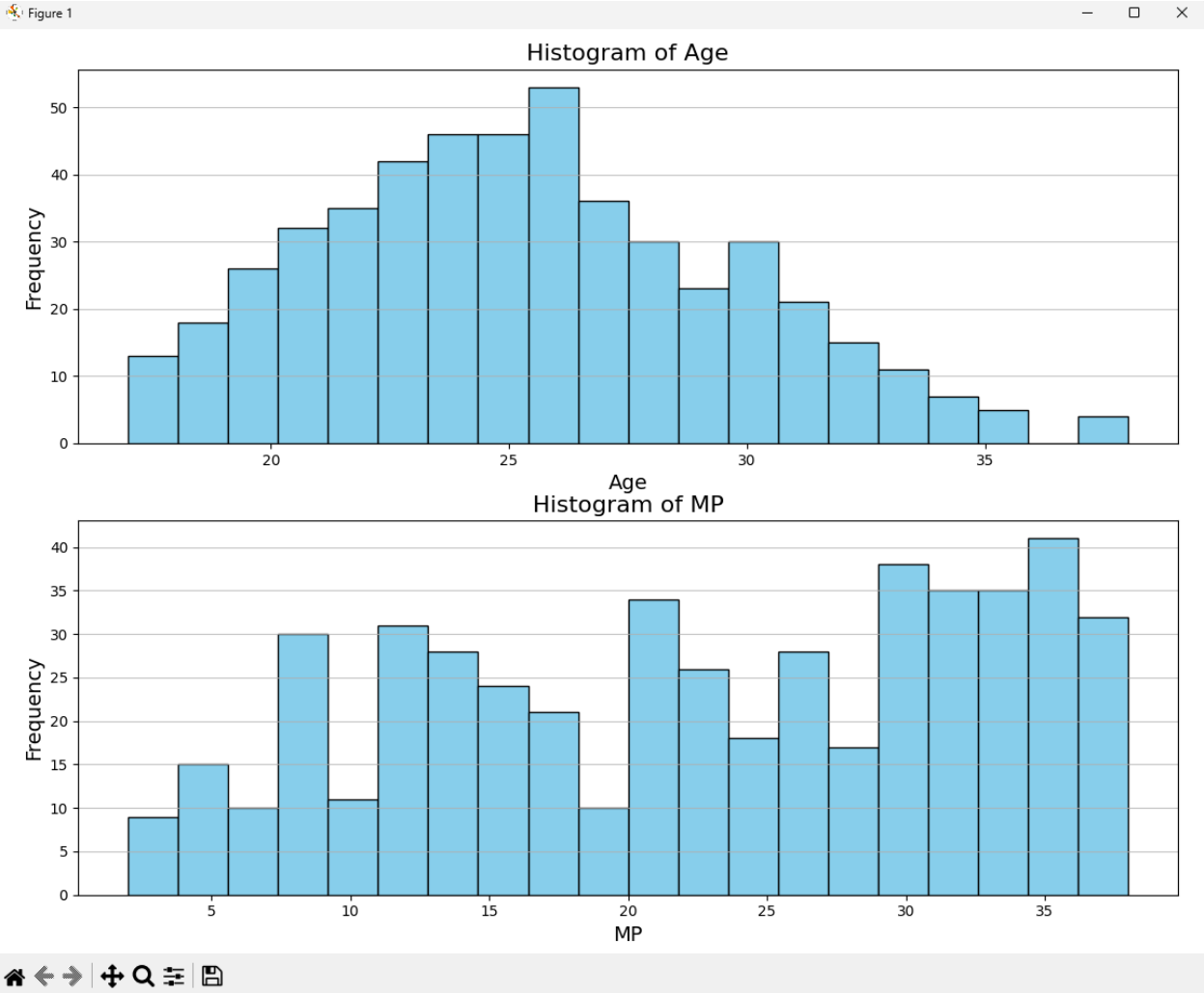


Figure 1
Histogram of Age
Frequency
Age
Histogram of MP
Frequency
MP

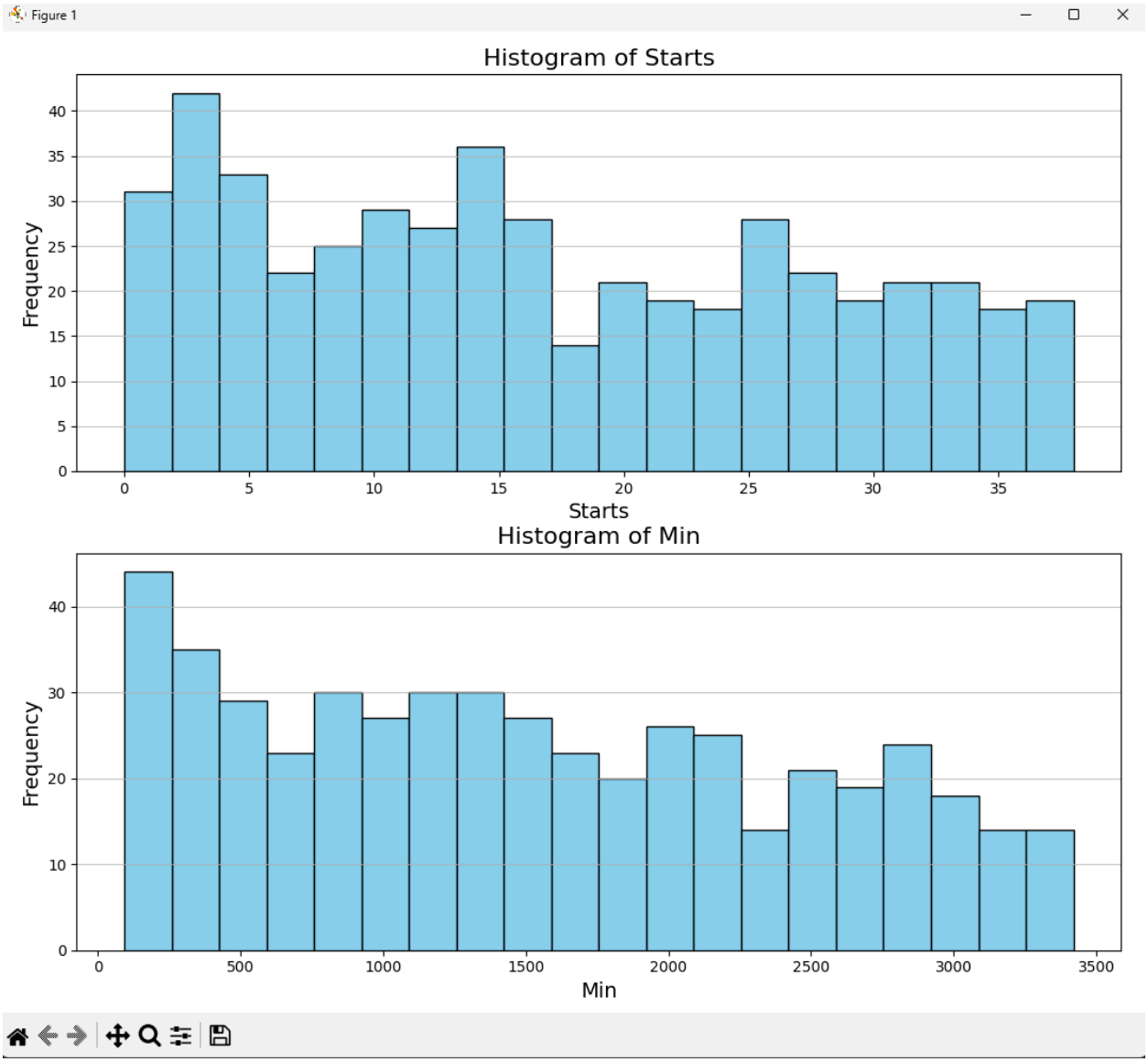
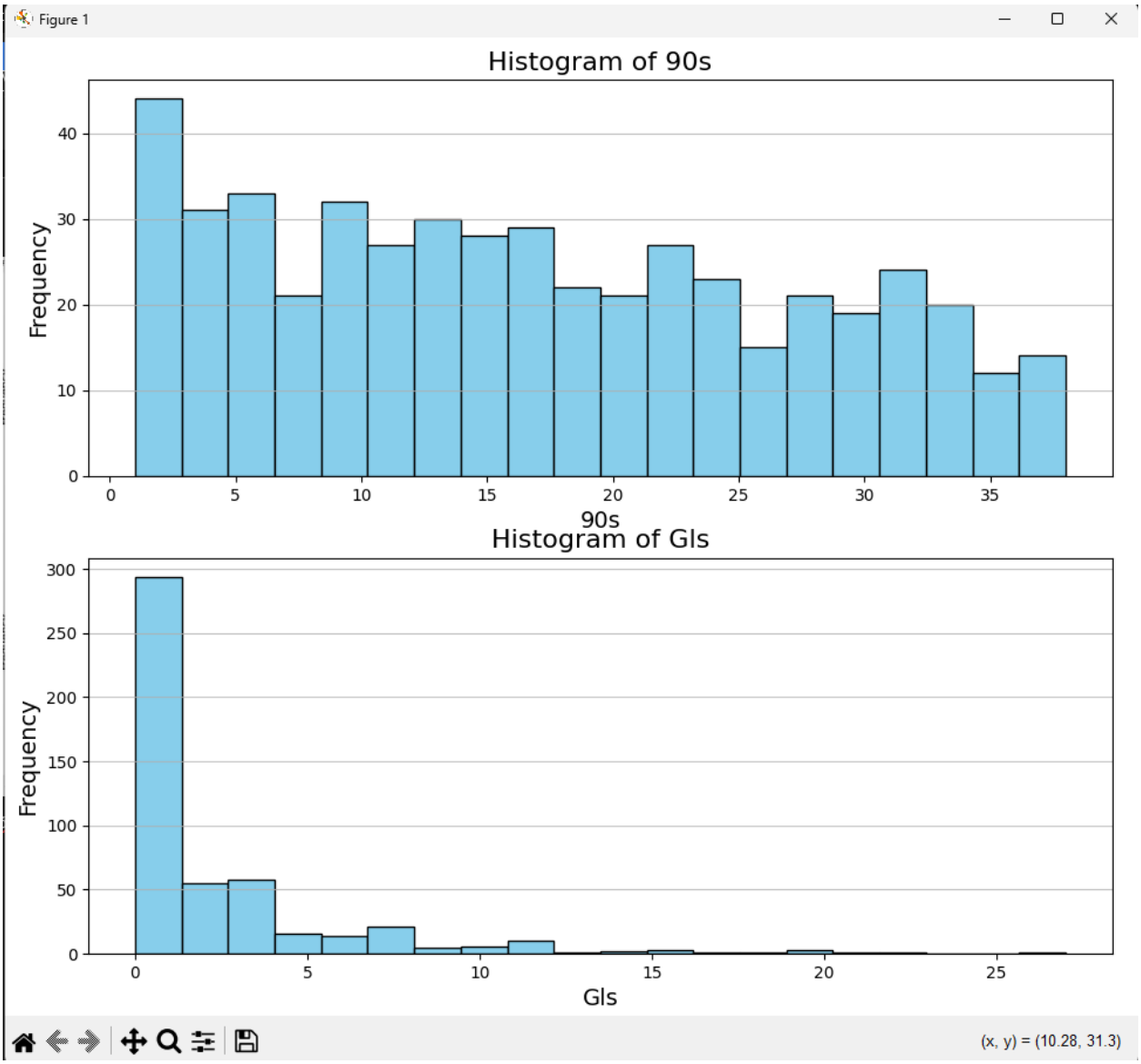


Figure 1
Histogram of Starts
Frequency
Starts
0
5
10
15
20
25
30
35
40
Histogram of Min
Frequency
Min
0
500
1000
1500
2000
2500
3000
3500
10
20
30
40

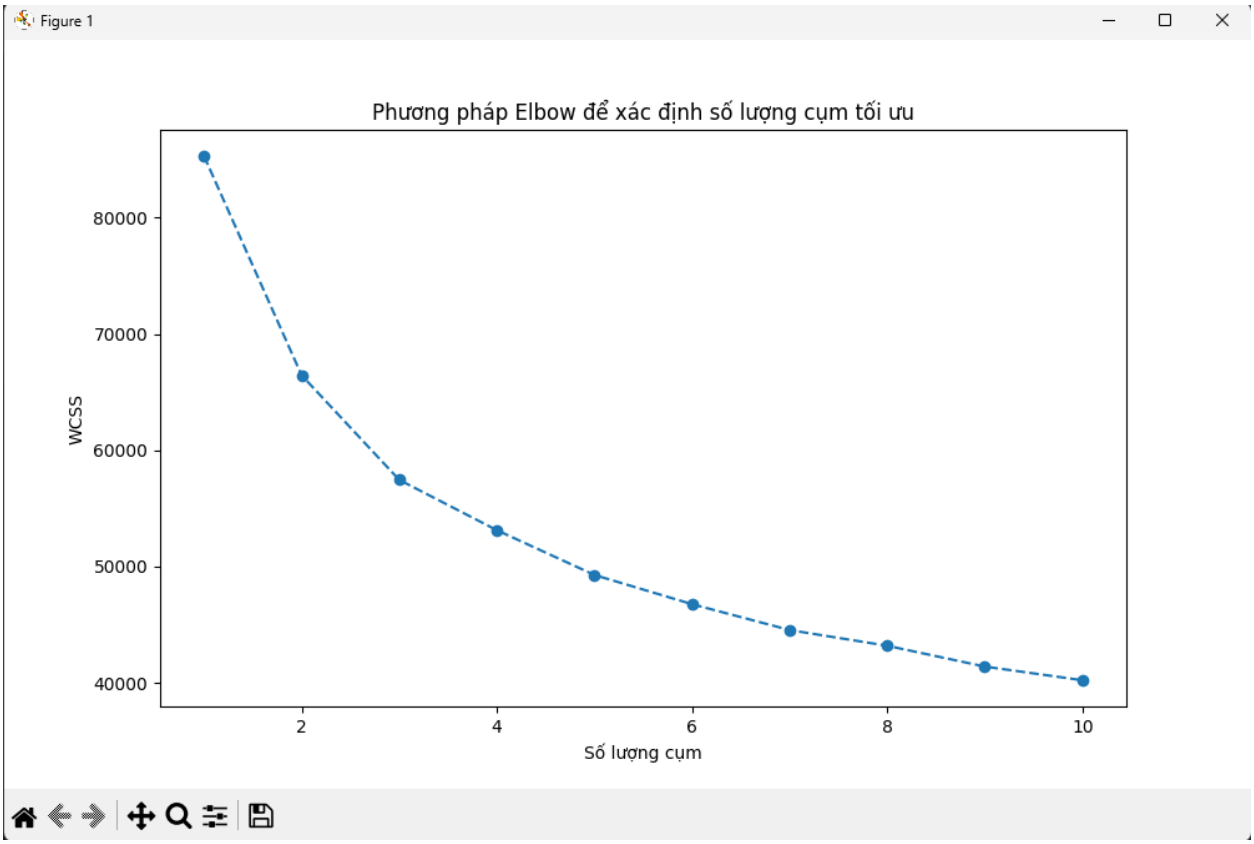


Bài 3:

Đây là code Kmeans và sử dụng lược đồ Elbow để xác định số lượng cụm tối ưu

```python
data = pd.read_csv('result.csv')
data['Min'] = data['Min'].str.replace(',', '').astype(int)
numeric_data = data.select_dtypes(include=['float64', 'int64'])
numeric_data = numeric_data.fillna(numeric_data.mean())

scaler = StandardScaler()
scaled_features = scaler.fit_transform(numeric_data)

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(scaled_features)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot( *args: range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel('Số lượng cụm')
plt.ylabel('WCSS')
plt.title('Phương pháp Elbow để xác định số lượng cụm tối ưu')
plt.show()
```

Đây là lược đồ Elbow



Ta có thể thấy số cụm tối ưu là 10.

Ta có thể phân loại cầu thủ thành 10 cụm cụ thể như sau:

1. **Cụm 1: Tiền đạo xuất sắc (High Scoring Forwards)**
    - Chỉ số ghi bàn và sút cao.
    - Có khả năng dứt điểm mạnh mẽ, thường chơi ở vai trò tiền đạo mũi nhọn hoặc tiền đạo cánh.
    - Cầu thủ trong nhóm này có thể là những người nổi bật trong việc ghi bàn và tạo cơ hội.
2. **Cụm 2: Tiền vệ tấn công (Attacking Midfielders)**
    - Chỉ số chuyền bóng, kiến tạo và kiểm soát bóng cao.
    - Thường là các cầu thủ ở vị trí tiền vệ tấn công, chuyên tạo ra các đường chuyền quyết định và dẫn dắt lối chơi.
    - Họ đóng vai trò cầu nối giữa hàng tiền vệ và hàng công, hỗ trợ trực tiếp cho các tiền đạo.
3. **Cụm 3: Trung vệ phòng ngự (Defensive Center Backs)**
    - Chỉ số tắc bóng, phá bóng và khả năng phòng ngự nổi bật.
    - Cầu thủ trong nhóm này chủ yếu là các trung vệ hoặc tiền vệ phòng ngự có nhiệm vụ bảo vệ khu vực phía trước khung thành.
    - Khả năng cắt bóng và đánh đầu thường cao, giúp ngăn chặn các đợt tấn công của đối thủ.
4. **Cụm 4: Hậu vệ cánh tốc độ (Fast Fullbacks)**
    - Chỉ số tốc độ, tắc bóng và hỗ trợ tấn công cao.
    - Họ là những hậu vệ cánh với khả năng lên công về thủ linh hoạt, có thể tham gia các tình huống tấn công biên.
    - Nhóm này thường có chỉ số về sức bền tốt vì họ di chuyển nhiều trong các trận đấu.
5. **Cụm 5: Tiền đạo phụ đa năng (Versatile Secondary Forwards)**

- Chỉ số tấn công khá tốt, nhưng không vượt trội như cụm tiền đạo chính.
- Thường đóng vai trò hỗ trợ các tiền đạo chính, tham gia nhiều vào các tình huống pressing và hỗ trợ lối chơi toàn diện.
- Có thể di chuyển linh hoạt trên hàng công hoặc lùi sâu hỗ trợ tiền vệ.

6. **Cụm 6: Tiền vệ trung tâm (Central Midfielders)**
   - Cân bằng giữa các chỉ số tấn công và phòng ngự.
   - Họ thường là những cầu thủ hoạt động rộng trên sân, có thể tham gia cả vào tấn công và phòng ngự.
   - Nhóm này có chỉ số chuyền bóng và kiểm soát bóng ổn định, nhưng không quá nổi bật về ghi bàn hoặc cản phá.
7. **Cụm 7: Hậu vệ đa năng (Versatile Defenders)**
   - Có chỉ số phòng ngự tốt, đồng thời có thể tham gia tấn công khi cần.
   - Các hậu vệ trong nhóm này có thể chơi ở cả vị trí trung vệ và hậu vệ cánh.
   - Họ thường được sử dụng linh hoạt, tuỳ thuộc vào yêu cầu chiến thuật của trận đấu.
8. **Cụm 8: Tiền vệ phòng ngự thuần túy (Pure Defensive Midfielders)**
   - Chỉ số phòng ngự rất cao, với khả năng cắt bóng, tắc bóng và giữ vị trí tốt.
   - Đây là các cầu thủ chuyên chơi ở vị trí tiền vệ phòng ngự, giúp bảo vệ hàng phòng ngự và ngăn chặn các đợt tấn công từ xa.
   - Họ có xu hướng hoạt động ở trung tâm sân và ít tham gia vào các tình huống tấn công.
9. **Cụm 9: Tiền vệ cánh nhanh nhẹn (Agile Wingers)**
   - Chỉ số tốc độ, rê bóng và chuyền bóng cao, nhưng không tập trung vào ghi bàn.

- Những cầu thủ này chủ yếu hoạt động ở biên, tạo các đường chuyền vào trong cho tiền đạo.
- Họ có thể lùi sâu để hỗ trợ phòng ngự, nhưng chủ yếu là những cầu thủ tấn công.

10. **Cụm 10: Thủ môn (Goalkeepers)**

- Chỉ số đặc trưng cho vị trí thủ môn, như cứu thua, phản xạ, và phát bóng.
- Nhóm này chỉ bao gồm các thủ môn, do họ có vai trò rất đặc thù và khác biệt với các cầu thủ ở vị trí khác.

Lí do nên chia thành 10 cụm: Việc phân chia 10 nhóm này sẽ giúp cung cấp một cái nhìn rõ hơn về vai trò của từng cầu thủ, đồng thời hỗ trợ huấn luyện viên và nhà phân tích trong việc xếp đội hình và chiến thuật.

Nhận xét về các cụm

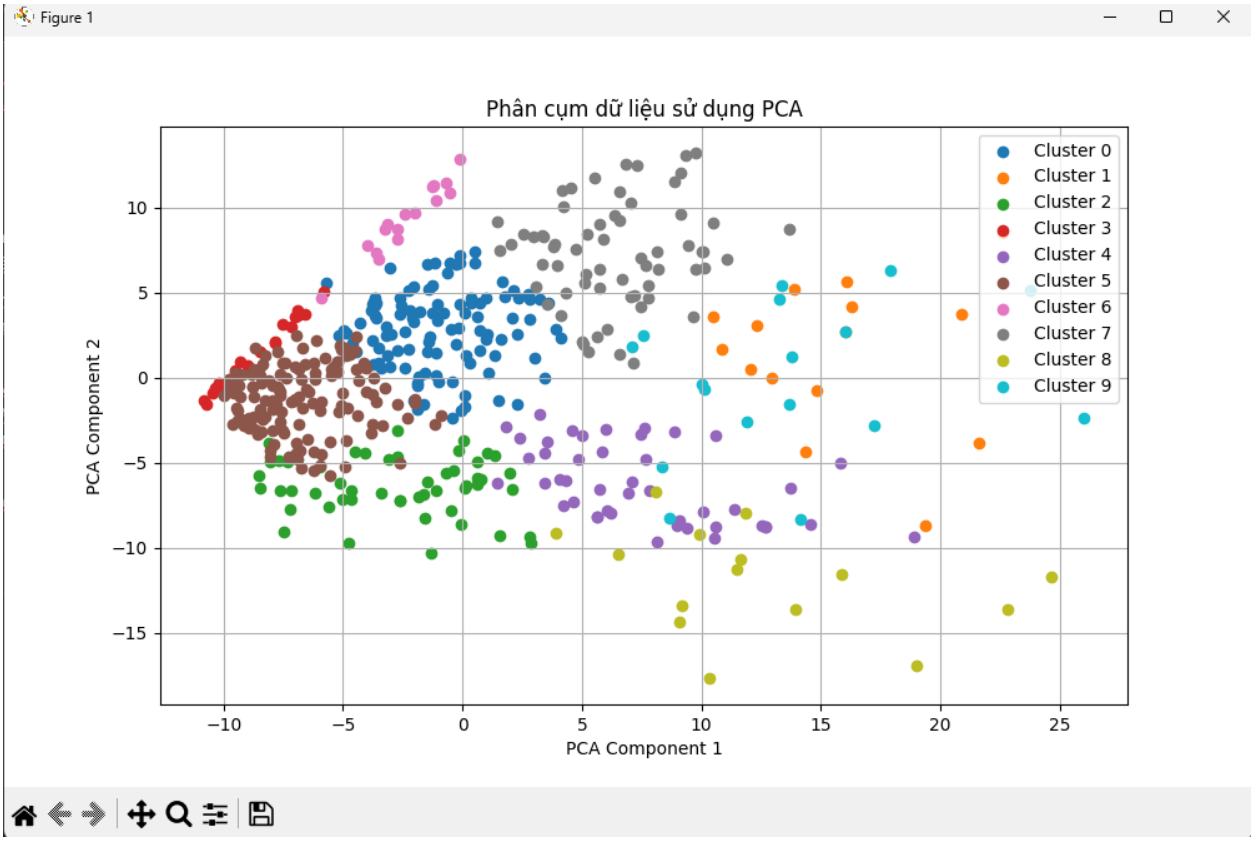
- **Độ chính xác của cụm**: Phân chia 10 cụm giúp chi tiết hóa vai trò của từng nhóm cầu thủ trong các khía cạnh khác nhau của trận đấu.
- **Ý nghĩa phân cụm**: Kết quả phân cụm giúp dễ dàng nhận diện các cầu thủ nổi bật trong từng vai trò và cũng cho phép so sánh nhanh hơn giữa các nhóm cầu thủ khác nhau.
- **Giới hạn của phân cụm K-means**: K-means dựa vào khoảng cách số học nên không phản ánh được sự phức tạp trong phong cách chơi của một số cầu thủ đa năng. Một số cầu thủ có thể không thuộc một cụm cố định mà thay đổi vai trò tùy theo chiến thuật của trận đấu.

Đây là code về thuật toán PCA và nên chia cầu thủ nào thuộc nhóm được đánh số từ 0 đến 9

```
data['Cluster'] = clusters
print(data[['Player', 'Cluster'] + numeric_data.columns.tolist()])

pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_features)

pca_df = pd.DataFrame(data=pca_result, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = clusters

plt.figure(figsize=(10, 6))
for i in range(optimal_clusters):
    plt.scatter(pca_df[pca_df['Cluster'] == i]['PCA1'],
                pca_df[pca_df['Cluster'] == i]['PCA2'],
                label=f'Cluster {i}')

plt.title('Phân cụm dữ liệu sử dụng PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.grid()
plt.show()
```

Đây là kết quả

```
"D:\Btap Code\Python\BaiTapLon\.venv\Scripts\python.exe" "D:\Btap Code\Python\Bai
                Player  Cluster  Age  ...  Aerial_Won  Aerial_Lost  Aerial_Won%
0      Aaron Cresswell        5   33  ...           6            3         66.7
1         Aaron Hickey        5   21  ...           1            9         10.0
2       Aaron Ramsdale        3   25  ...           0            0          NaN
3         Aaron Ramsey        5   20  ...           4            5         44.4
4    Aaron Wan-Bissaka        0   25  ...          21           19         52.5
..                 ...      ...  ...  ...         ...          ...          ...
488      Yves Bissouma        7   26  ...          14           15         48.3
489       Zeki Amdouni        4   22  ...          15           56         21.1
490        Álex Moreno        5   30  ...           4           10         28.6
491     Đorđe Petrović        3   23  ...           5            0        100.0
492   Łukasz Fabiański        3   38  ...           2            0        100.0
```

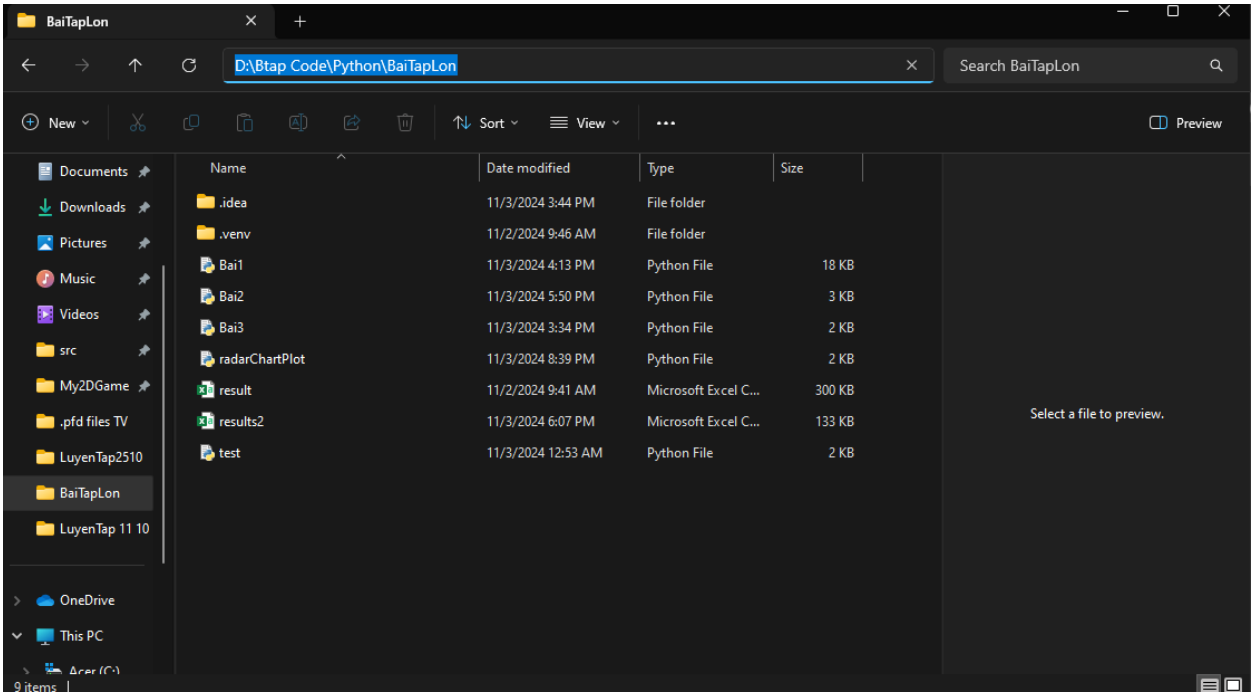


Đây là code radarChartPlot.py

```python
import argparse
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from math import pi

data = pd.read_csv( filepath_or_buffer: 'result.csv', encoding='utf-8')
1 usage
def create_radar_chart(player1_data, player2_data, attributes, player1_name, player2_name):
    num_vars = len(attributes)
    angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)]
    angles += angles[:1]
    fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))

    player1_stats = player1_data[attributes].values.flatten().tolist()
    player1_stats += player1_stats[:1]
    ax.plot( *args: angles, player1_stats, linewidth=1, linestyle='solid', label=player1_name)
    ax.fill( *args: angles, player1_stats, alpha=0.25)

    player2_stats = player2_data[attributes].values.flatten().tolist()
    player2_stats += player2_stats[:1]
    ax.plot( *args: angles, player2_stats, linewidth=1, linestyle='solid', label=player2_name)
    ax.fill( *args: angles, player2_stats, alpha=0.25)

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)

    plt.title(f"Comparison between {player1_name} and {player2_name}")
    plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1))

    plt.show()
```

```python
parser = argparse.ArgumentParser(description="Radar chart comparison between two players.")
parser.add_argument( *name_or_flags: "--p1", type=str, required=True, help="Name of the first player")
parser.add_argument( *name_or_flags: "--p2", type=str, required=True, help="Name of the second player")
parser.add_argument( *name_or_flags: "--Attribute", type=str, required=True, help="Comma-separated list of attributes to compare")

args = parser.parse_args()

player1_name = args.p1
player2_name = args.p2
attributes = [attr.strip() for attr in args.Attribute.split(",")]

player1_data = data[data["Player"] == player1_name]
player2_data = data[data["Player"] == player2_name]

if player1_data.empty or player2_data.empty:
    print("One or both players not found in the dataset. Please check the names.")
else:
    create_radar_chart(player1_data, player2_data, attributes, player1_name, player2_name)
```

Đầu tiên dẫn Cmd đến file chứa code




```
D:\> cd Btap Code\Python\BaiTapLon

D:\Btap Code\Python\BaiTapLon>
```

Giả sử ta chạy thử 2 cầu thủ như sau:

python radarChartPlot.py --p1 "Aaron Cresswell" --p2 "Aaron Hickey" --Attribute "Gls,Fls,Aerial_Won"

```
D:\Btap Code\Python\BaiTapLon>python radarChartPlot.py --p1 "Aaron Cresswell" --p2 "Aaron Hickey" --Attribute "Gls,Fls,A
erial_Won"
```

Đây là kết quả

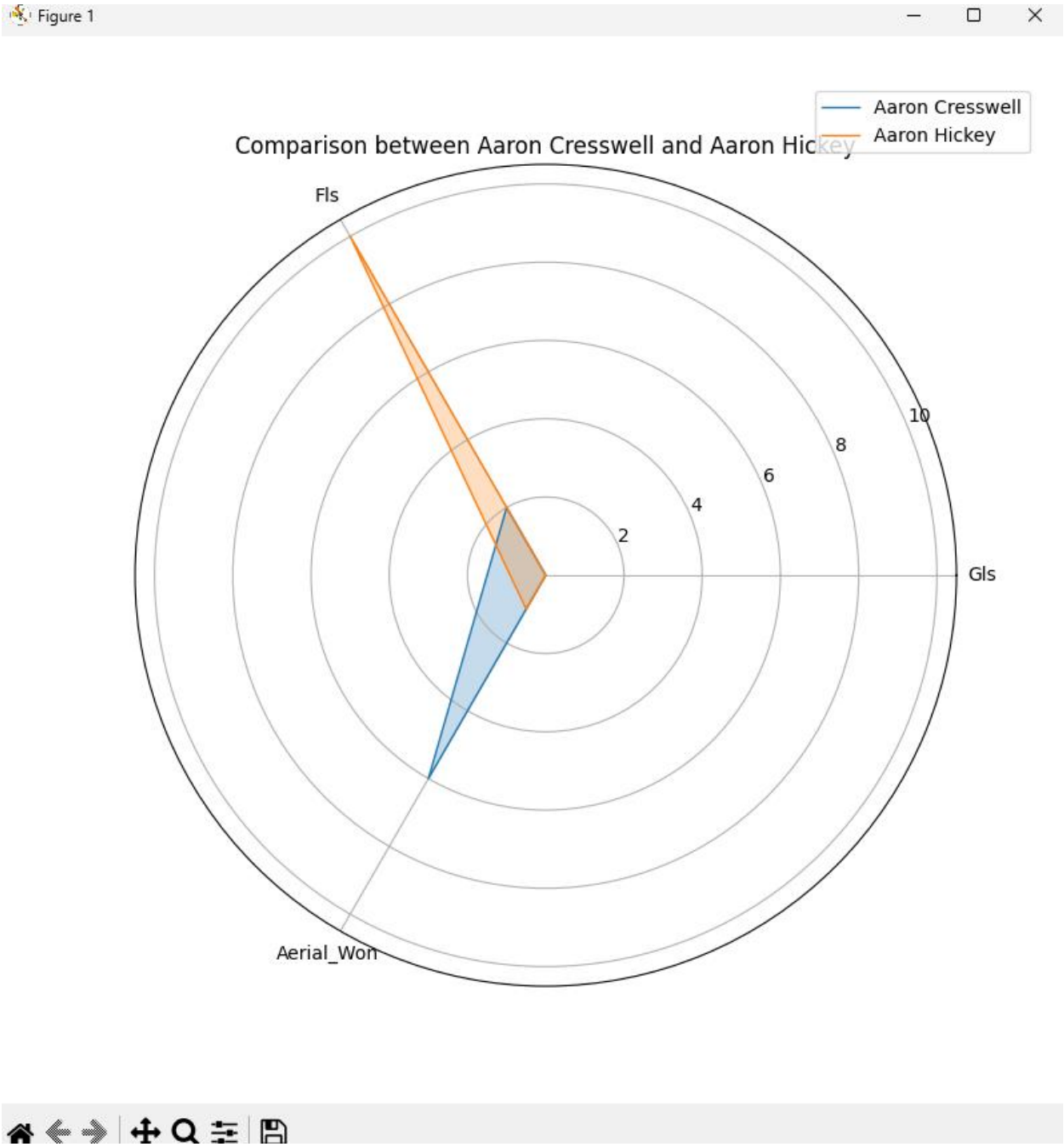


Figure 1
Aaron Cresswell
Aaron Hickey
Comparison between Aaron Cresswell and Aaron Hickey
Fls
Gls
Aerial_Won
2
4
6
8
10