Báo cáo assignment 1

Họ và tên: Dương Phan Bảo Linh
Mã sinh viên: B22DCCN485

Part I:

Requirements: BeautifulSoup4, lxml lib, pandas và requests module.

Các chỉ số em đã lấy được và có tên biến lần lượt như sau:

+ Player name: player

+ Nation: nationality

+ Team: team

+ Pos: position

+ Age: age

+ Born: birth_year

+ Playing time

    Matches Played: games

    Starts: games_starts

    Minutes: minutes

+ Performance:

    Goals: goals

    non-Penalty Goals: goals_pens

    Ast: assists

    Penalty Goals: pens_made

    CrdY: cards_yellow, CrdR: cards_red

+ Expected:

xG: xg

npxG: npxg

xAG: xg_assist

+ Progression:

PrgC: progressive_carries

PrgP: progressive_passes

PrgR: progressive_passes_received

+ Per 90 mins:

Gls: goals_per90

Ast: assists_per90

G+A: goals_assists_per90

G-PK: goals_pens_per90

G+A-PK: goals_assists_pens_per90

xG: xg_per90

xAG: xg_assist_per90

xG+xAG: xg_xg_assist_per90

npxG: npxg_per90

npxG+xAG: npxg_xg_assist_per90

+ Goalkeeping:

- Performance:

GA: gk_goals_against

GA90: gk_goals_against_per90

SoTA: gk_shots_on_target_against

Saves: gk_saves

Save%: gk_save_pct

W: gk_wins

D: gk_ties

L: gk_losses

CS: gk_clean_sheets

CS%: gk_clean_sheets_pct

- Penalty Kicks

  PKatt: gk_pens_att

  PKA: gk_pens_allowed

  PKsv: gk_pens_saved

  PKm: gk_pens_missed

  Save%: gk_pens_save_pct

+ Shooting:

- Standard

  Gls: goals

  Sh: shots

  SoT: shots_on_target

  SoT%: shots_on_target_pct

  Sh/90: shots_per90

  SoT/90: shots_on_target_per90

  G/Sh: goals_per_shot

  G/SoT: goals_per_shot_on_target

  Dist: average_shot_distance

  FK: shots_free_kicks

  PK: pens_made

  PKatt: pens_att

- Expected

  xG: xg

npxG/Sh: npxg_per_shot

G-xG: xg_net

np:G-xG: npxg_net

+ Passing

Cmp: passes_completed

Att: passes

Cmp%: passes_pct

TotDist: passes_total_distance

PrgDist: passes_progressive_distance

Cmp: passes_completed_short

Att: passes_short

Cmp%: passes_pct_short

Cmp: passes_completed_medium

Att: passes_medium

Cmp%: passes_pct_medium

Cmp: passes_completed_long

Att: passes_long

Cmp%: passes_pct_long

xA: pass_xa

A-xAG: xg_assist_net

KP: assisted_shots

1/3: passes_into_final_third

PPA: passes_into_penalty_area

CrsPA: crosses_into_penalty_area

+ Pass types

Live: passes_live

Dead: passes_dead

FK: passes_free_kicks

TB: through_balls

Sw: passes_switches

Crs: crosses

TI: throw_ins

CK: corner_kicks

In: corner_kicks_in

Out: corner_kicks_out

Str: corner_kicks_straight

Off: passes_offsides

Blocks: passes_blocked

+ Goal and Shot Creation:

SCA: sca

SCA90: sca_per90

PassLive: sca_passes_live

PassDead: sca_passes_dead

TO: sca_take_ons

Sh: sca_shots

Fld: sca_fouled

Def: sca_defense

GCA: gca

GCA90: gca_per90

PassLive: gca_passes_live

PassDead: gca_passes_dead

TO: gca_take_ons

Sh: gca_shots

Fld: gca_fouled

Def: gca_defense

+ Defensive Actions:

Tkl: tackles

TklW: tackles_won

Def 3rd: tackles_def_3rd

Mid 3rd: tackles_mid_3rd

Att 3rd: tackles_att_3rd

Tkl: challenge_tackles

Att: challenges

Tkl%: challenge_tackles_pct

Lost: challenges_lost

Blocks: blocks

Sh: blocked_shots

Pass: blocked_passes

Int: interceptions

Tkl+Int: tackles_interceptions

Clr: clearances

Err: errors

+ Possession

Touches: touches

Def Pen: touches_def_pen_area

Def 3rd: touches_def_3rd

Mid 3rd: touches_mid_3rd

Att 3rd: touches_att_3rd

Att Pen: touches_att_pen_area

Live: touches_live_ball

Att: take_ons

Succ: take_ons_won

Succ%: take_ons_won_pct

Tkld: take_ons_tackled

Tkld%: take_ons_tackled_pct

Carries: carries

TotDist: carries_distance

PrgDist: carries_progressive_distance

1/3: carries_into_final_third

CPA: carries_into_penalty_area

Mis: miscontrols

Dis: dispossessed

Rec: passes_received

+ Playing time:

Mn/MP: minutes_per_game

Mn/Start: minutes_per_start

Compl: games_complete

Subs: games_subs

Mn/Sub: minutes_per_sub

unSub: unused_subs

PPM: points_per_game

onG: on_goals_for

onGA: on_goals_against

On-Off: plus_minus_wowy

onxG: on_xg_for

onxGA: on_xg_against

Fls: fouls

Fld: fouled

Off: offsides

OG: own_goals

Recov: ball_recoveries

Won: aerials_won

Lost: aerials_lost

Won%: aerials_won_pct

Kết quả được lưu tại **./output/result.csv**

Part II:

+ Top 3 điểm cầu thủ cao nhất và thấp nhất ở mỗi chỉ số:

Kết quả được lưu tại ./output/top_3_best_players.txt

./output/top_3_worst_players.txt

+ Tìm trung vị của mỗi chỉ số:

Kết quả mean, median, std được lưu tại **./output/result2.csv**

Kết quả histogram của trung bình được lưu tại **./images/all (hoặc <team>)**

Trong đó team là tên đội truyển

- Xét đoạn code sau:

```
f = open('./output/best_team_in_each_category.txt', 'w',
encoding='utf-8')
f.write('Best team in each category:\n\n'.upper())
freq = {}
for category in categories:
```

```
    max_mean = 0
    best_team = ''
    for team in sorted(teams):
        team_players = list(filter(lambda x: x.stats['team'] ==
team, players))
        mean, _, _ = calculate_mean_median_std(team_players,
category)
        if mean > max_mean:
            max_mean = mean
            best_team = team

    f.write(f'Best team in {category: <35}: {best_team: <20} : mean
= {max_mean: .4f}\n')
    if best_team in freq:
        freq[best_team] += 1
    else:
        freq[best_team] = 1
f.close()

print(max(freq, key=freq.get))
```

Trong mỗi chỉ số chúng ta tìm tần suất xuất hiện đội tuyển nhiều nhất:

```
if best_team in freq:
        freq[best_team] += 1
    else:
        freq[best_team] = 1
```

Và in ra kết quả:

```
print(max(freq, key=freq.get))
```

Kết quả trả về: **Manchester City**

Dựa vào đây với mùa giải được xét, đội Manchester City là đội có phong độ tốt nhất!

Part 3:

Requirements: BeautifulSoup4, pandas, seaborn, scikit-learn.

- Dữ liệu được đọc từ tệp CSV (result.csv) và các cột cần thiết được chọn lọc.
- Dữ liệu được chuẩn hóa bằng StandardScaler để đảm bảo rằng các đặc trưng có cùng đơn vị đo lường.

```
# Filter data to include only the selected columns
data_filtered = data[attributes].dropna()


# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_filtered)
```

- Hàm find_optimal_clusters sử dụng phương pháp "Elbow" để xác định số lượng cụm tối ưu cho thuật toán K-Means. Phương pháp này tính toán tổng bình phương sai số (SSE) cho các số lượng cụm khác nhau và vẽ đồ thị để tìm "elbow" của đồ thị.

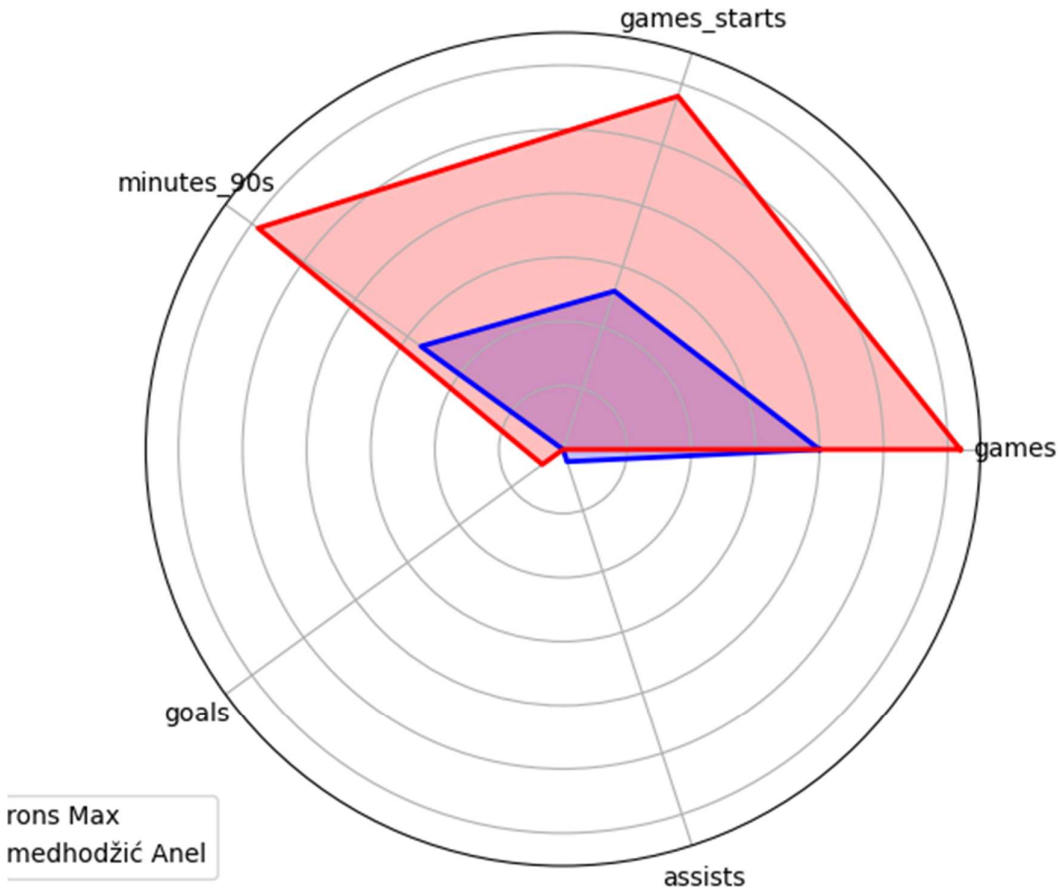
```
def find_optimal_clusters(data, max_k):
    iters = range(1, max_k+1)
    sse = []
    for k in iters:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        sse.append(kmeans.inertia_)
```

```
    plt.figure(figsize=(8, 6))
    plt.plot(iters, sse, marker='o')
    plt.xlabel('Cluster Centers')
    plt.ylabel('SSE')
    plt.title('Elbow Method For Optimal k')
    plt.savefig('./images/elbow.png')
    plt.show()
```

- Sau khi xác định số lượng cụm tối ưu, thuật toán K-Means được áp dụng để phân cụm dữ liệu.
- Kết quả phân cụm được thêm vào dữ liệu gốc.
- Dữ liệu được giảm chiều xuống 2D bằng phương pháp PCA (Phân tích thành phần chính).

```
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)
data['pca1'] = data_pca[:, 0]
data['pca2'] = data_pca[:, 1]
```

- Các cụm được trực quan hóa trên mặt phẳng 2D bằng biểu đồ phân tán.
- Hàm radar_chart tạo biểu đồ radar để so sánh các đặc trưng của hai cầu thủ cụ thể.
- Biểu đồ này hiển thị các đặc trưng của hai cầu thủ trên cùng một biểu đồ để dễ dàng so sánh.
  Ví dụ CLI:
  python -u ".\code\src\part_three.py" --p1 "Aarons Max" --p2 "Ahmedhodžić Anel" --Attribute games,games_starts,minutes_90s,goals,assists
  Ta được kết quả:



Part 4:

Bao gồm các bước:

Scrap data từ: **https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/2023-2024**

Kết quả được lưu tại: **./code/output/result4.csv**