HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

—————o0o—————

BÁO CÁO BÀI TẬP LỚN PYTHON

Giảng Viên: **Kim Ngọc Bách**
Sinh viên thực hiện: **Nguyễn Khánh Tùng**
Mã sinh viên: **B23DCCD096**
Lớp: **D23CQCE06-B**
Niên khóa: **2023-2028**
Hệ đào tạo: **Đại học chính quy**

Hà Nội, 5/2025

Chương 1

Assignment 1:

1.1 Câu 1: Thu thập chỉ số của các cầu thủ được yêu cầu trong đề:

- Các lưu ý trong mã nguồn thực hiện yêu cầu của đề bài:

- Mã nguồn sử dụng Selenium thay vì Request, vì thế cần cài đặt Driver phù hợp với trình duyệt tương ứng với máy tính dùng để chạy chương trình (Ở đây sử dụng Microsoft Edge):

```
# === Cài đặt Edge WebDriver ===
edge_options = EdgeOptions()
edge_options.add_argument("--headless")
edge_options.add_argument("--disable-gpu")
service = EdgeService(executable_path="C:\Webdrivers\msedgedriver.exe")
driver = webdriver.Edge(service=service, options=edge_options)
```

- Vì có nhiều bảng từ nhiều URL cần lấy dữ liệu mà các bảng có các cột thống kê trùng nhau (quốc tịch, tuổi tác) nên ta cần phải xử lý loại bỏ trùng lặp, đảm bảo file CSV đầu ra các thống kê trùng lặp chỉ xuất hiện 1 lần duy nhất:

```
# Xử lí các thông số bị trùng nhau giữa các bảng trong các URL được scrape:
for name, df in stat_tables.items():
    if name == "standard":
        continue

    df = df.loc[:, ~df.columns.duplicated()]  # Loại bỏ tất cả các cột trùng nhau
    df = df.rename(columns={"Squad": "Team", "Pos": "Position"})

    if 'Player' in df.columns:
        # Kiểm tra xem nên giữ lại những cột nào
        key_cols = ['Player']
        if 'Team' in df.columns and 'Team' in merged_df.columns:
            key_cols.append('Team')
            join_cols = ['Player', 'Team']
        else:
            join_cols = ['Player']

        # Loại bỏ các cột đã có trong merged_df và gộp vào
        drop_cols = [col for col in df.columns if col in merged_df.columns and col not in join_cols]
        df = df.drop(columns=drop_cols)
        merged_df = pd.merge(merged_df, df, on=join_cols, how='left')

        # Xóa các cột trùng sau khi gộp
        merged_df = merged_df.loc[:, ~merged_df.columns.duplicated()]
    else:
        continue
```

- Kết quả: Thu được file CSV có dạng như sau:

	Player	Nation	Position	Team	Age	Born	MP	Starts	Min	90s	Gls	Ast	G+A	G-PK	PK	PKatt	CrdY	CrdR	xG	npxG	xAG
1	Aaron Ramsdale	eng ENG	GK	Southampton	26-355	1998	27	27	2430	27.0	0	0	0	0	0	0	2	0	0.0	0.0	0.0
2	Aaron Cresswell	eng ENG	DF	West Ham	35-140	1989	14	7	589	6.5	0	0	0	0	0	0	2	0	0.1	0.1	1.1
3	Aaron Wan-Bissaka	eng ENG	DF	West Ham	27-159	1997	32	31	2794	31.0	2	2	4	2	0	0	1	0	1.2	1.2	2.9
4	Abdoulaye Doucouré	ml MLI	MF	Everton	32-123	1993	30	29	2425	26.9	3	1	4	3	0	0	5	1	3.9	3.9	2.3
5	Abdukodir Khusanov	uz UZB	DF	Manchester City	21-064	2004	6	6	503	5.6	0	0	0	0	0	0	1	0	0.0	0.0	0.1
6	Abdul Fatawu Issahaku	gh GHA	FW	Leicester City	21-057	2004	11	6	579	6.4	0	2	2	0	0	0	0	0	0.4	0.4	1.6
7	Adam Lallana	eng ENG	MF	Southampton	36-359	1988	14	5	361	4.0	0	2	2	0	0	0	4	0	0.2	0.2	0.9
8	Adam Armstrong	eng ENG	FW,MF	Southampton	28-083	1997	20	15	1248	13.9	2	2	4	2	0	1	4	0	3.3	2.3	1.2
9	Adam Webster	eng ENG	DF	Brighton	30-120	1995	11	8	617	6.9	0	0	0	0	0	0	0	0	0.0	0.0	0.5
10	Adam Smith	eng ENG	DF	Bournemouth	34-005	1991	22	17	1409	15.7	0	0	0	0	0	0	6	0	0.7	0.7	0.3
11	Adam Wharton	eng ENG	MF	Crystal Palace	20-336	2004	19	15	1258	14.0	0	2	2	0	0	0	2	0	0.3	0.3	3.0
12	Adama Traoré	es ESP	FW,MF	Fulham	29-099	1996	33	16	1592	17.7	2	6	8	2	0	0	3	0	3.9	3.9	4.7
13	Albert Grønbaek	dk DEN	FW,MF	Southampton	23-346	2001	4	2	143	1.6	0	0	0	0	0	0	0	0	0.1	0.1	0.0
14	Alejandro Garnacho	ar ARG	MF,FW	Manchester Utd	20-307	2004	33	22	2056	22.8	5	1	6	5	0	0	2	0	7.0	7.0	3.6
15	Alex Palmer	eng ENG	GK	Ipswich Town	28-267	1996	11	11	990	11.0	0	0	0	0	0	0	2	0	0.0	0.0	0.0
16	Alex Iwobi	ng NGA	FW,MF	Fulham	29-001	1996	35	33	2796	31.1	9	6	15	9	0	0	1	0	4.6	4.6	6.9
17	Alex Scott	eng ENG	MF	Bournemouth	21-256	2003	18	7	709	7.9	0	0	0	0	0	0	2	0	0.7	0.7	0.8
18	Alex McCarthy	eng ENG	GK	Southampton	35-152	1989	5	5	450	5.0	0	0	0	0	0	0	0	0	0.0	0.0	0.0
19	Alexander Isak	se SWE	FW	Newcastle Utd	25-225	1999	31	31	2487	27.6	22	6	28	19	3	3	1	0	19.0	16.6	4.0
20	Alexis Mac Allister	ar ARG	MF	Liverpool	26-131	1998	33	30	2553	28.4	5	4	9	5	0	0	6	0	2.8	2.8	4.4
21	Ali Al Hamadi	iq IRQ	FW	Ipswich Town	23-064	2002	11	0	134	1.5	0	0	0	0	0	0	3	0	0.4	0.4	0.1
22	Alisson	br BRA	GK	Liverpool	32-214	1992	24	24	2148	23.9	0	0	0	0	0	0	0	0	0.0	0.0	0.6
23	Alphonse Areola	fr FRA	GK	West Ham	32-066	1993	23	22	1990	22.1	0	0	0	0	0	0	0	0	0.0	0.0	0.0
24	Amad Diallo	ci CIV	FW,MF	Manchester Utd	22-297	2002	22	17	1594	17.7	6	6	12	6	0	0	3	0	4.1	4.1	3.9
25	Amadou Onana	be BEL	MF	Aston Villa	23-261	2001	23	17	1378	15.3	3	0	3	3	0	0	4	0	2.1	2.1	0.3
26	Andreas Pereira	br BRA	MF	Fulham	29-123	1996	31	23	1879	20.9	2	4	6	2	0	1	7	0	3.5	2.7	4.5
27	Andrew Robertson	sct SCO	DF	Liverpool	31-054	1994	31	27	2308	25.6	0	0	0	0	0	0	3	1	1.0	1.0	4.1
28	André	br BRA	MF	Wolves	23-292	2001	30	28	2249	25.0	0	0	0	0	0	0	7	0	0.4	0.4	0.7
29	André Onana	cm CMR	GK	Manchester Utd	29-032	1996	33	33	2970	33.0	0	0	0	0	0	0	0	0	0.0	0.0	0.2
30	Andrés García	es ESP	DF,MF	Aston Villa	22-086	2003	7	5	318	3.5	0	0	0	0	0	0	0	0	0.0	0.0	0.2
31	Andy Irving	sct SCO	MF	West Ham	24-356	2000	10	1	166	1.8	0	0	0	0	0	0	2	0	0.1	0.1	0.2

1.2 Câu 2:

1.2.1 Yêu cầu thứ 1 + 2: Identify the top 3 players with the highest and lowest scores for each statistic. Save result to a file name 'top_3.txt'Find the median for each statistic. Calculate the mean and standard deviation for each statistic across all players and for each team. Save the results to a file named 'results2.csv' with the following format:

Hàm để lấy top 3 cầu thủ với điểm số cao nhất và thấp nhất cho mỗi chỉ số:

```
# Top 3 cao nhất - thấp nhất cho từng chỉ số:
top3_lines = []
for col in numeric_df.columns:
    if numeric_df[col].dropna().empty:
        continue
    top3_lines.append(f"\n--- Top 3 HIGH for {col} ---")
    top3_lines.extend(df_combined.nlargest(3, col)[['Player', 'Team', col]].to_string(index=False).split('\n'))
    top3_lines.append(f"\n--- Top 3 LOW for {col} ---")
    top3_lines.extend(
        df_combined.nsmallest(3, col)[['Player', 'Team', col]]
        .to_string(index=False).split('\n')
    )
```

Tính toán median, mean và standard derivation cho các chỉ số của các cầu thủ cho mỗi đội và lưu file đúng với yêu cầu của bài toán: Sử dụng các hàm median(), mean(), std() có sẵn trong thư viện numpy:

```
# Tính toán median, mean và standard derivation cho các đội:
def compute_team_stats(df_subset):
    stats = {}
    for col in numeric_df.columns:
        stats[f"Median of {col}"] = df_subset[col].median()
        stats[f"Mean of {col}"] = df_subset[col].mean()
        stats[f"Std of {col}"] = df_subset[col].std()
    return stats
```

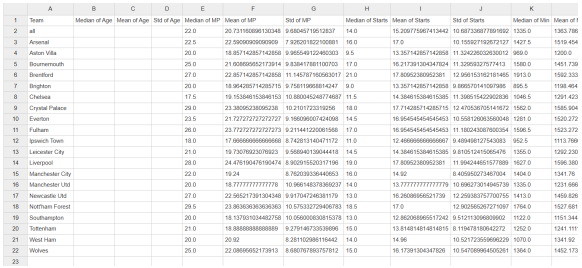
Lưu ra các file: result2.csv và top_3.txt:

```
rows = []
overall_stats = compute_team_stats(numeric_df)
overall_stats["Team"] = "all"
rows.append(overall_stats)
for team, group in df_combined.groupby("Team"):
    stats = compute_team_stats(group[numeric_df.columns])
    stats["Team"] = team
    rows.append(stats)

# Lưu file
results2_df = pd.DataFrame(rows)
results2_df = results2_df[['Team'] + [col for col in results2_df.columns if col != 'Team']]
results2_df.to_csv("results2.csv", index=False, encoding="utf-8")
print("Saved to 'top_3.txt' and 'results2.csv'")
```

Kết quả:

```
--- Top 3 HIGH for Gls ---
         Player             Team  Gls
 Mohamed Salah        Liverpool   28
Alexander Isak    Newcastle Utd   22
Erling Haaland Manchester City   21

--- Top 3 LOW for Gls ---
            Player             Team  Gls
     Aaron Ramsdale      Southampton    0
    Aaron Cresswell         West Ham    0
Abdukodir Khusanov Manchester City    0
```

Hình 1.1: File top_3.txt



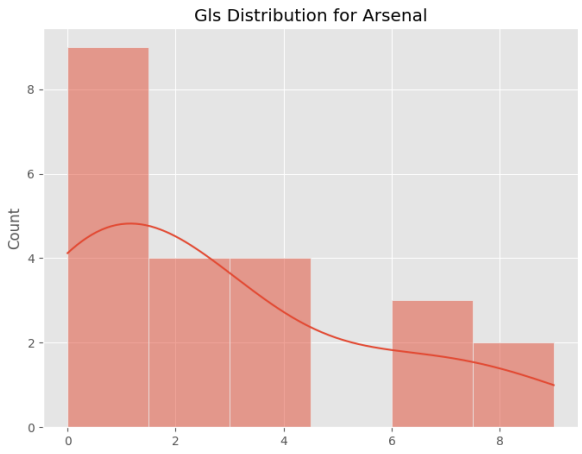
	A	B	C	D	E	F	G	H	I	J	K	
1	Team	Median of Age	Mean of Age	Std of Age	Median of MP	Mean of MP	Std of MP	Median of Starts	Mean of Starts	Std of Starts	Median of Min	Mean of M
2	all				22.0	20.731908996130348	9.680457195123837	14.0	15.209775967413442	10.687336877891692	1335.0	1363.786
3	Arsenal				22.5	22.590909090909	7.926201822100881	16.0	17.0	10.155927192672127	1427.5	1519.454
4	Aston Villa				20.0	18.857142857142858	9.965546122460303	9.5	13.357142857142858	11.324228032630012	969.0	1200.0
5	Bournemouth				25.0	21.608695652173914	9.838417788110070	17.0	16.217391304347824	11.329932375774	1580.0	1451.739
6	Brentford				27.0	22.857142857142858	11.145787165093017	21.0	17.809523809523811	12.996153116218145	1913.0	1592.333
7	Brighton				20.0	18.964285714285715	9.758119958814247	9.0	13.357142857142858	9.8955701410979886	895.5	1198.464
8	Chelsea				17.5	19.153846153846153	10.880045248774687	11.5	14.384615384615385	11.368515422902836	1046.5	1291.423
9	Crystal Palace				29.0	23.380952380952381	10.210172331919256	18.0	17.714285714285715	12.470536705141672	1582.0	1585.904
10	Everton				23.5	21.727272727272727	9.196096007424068	14.5	16.954545454545453	10.558126063560048	1281.0	1520.272
11	Fulham				26.0	23.772727272727273	9.211441220061568	17.0	16.954545454545453	11.182438076500354	1598.5	1523.272
12	Ipswich Town				18.0	17.666666666666668	8.742813140471172	11.0	12.466666666666667	9.489496127543083	952.5	1113.769
13	Leicester City				21.0	19.730769230769223	9.568940130044418	14.5	14.384615384615385	9.810512415065476	1355.0	1292.230
14	Liverpool				28.0	24.476190476190474	8.902915520317196	19.0	17.809523809523811	11.994244651577889	1627.0	1596.380
15	Manchester City				22.0	19.24	8.762039338446653	16.0	14.92	8.405662273407204	1404.0	1341.76
16	Manchester Utd				20.0	18.777777777777778	10.966148378336237	14.0	13.777777777777779	10.699273014945739	1335.0	1231.666
17	Newcastle Utd				27.0	22.565217391304348	9.917047246381179	13.0	16.260869565217391	12.259383757700755	1413.0	1459.826
18	Nottham Forest				29.5	23.863636363636363	10.575332729406783	18.5	17.0	12.902565267271097	1764.0	1527.681
19	Southampton				20.0	18.137931034482758	10.056900830815378	13.0	12.862068965517242	9.512113096809902	1122.0	1151.344
20	Tottenham				21.0	18.888888888888889	8.279146733539896	15.0	13.814814814814815	8.119478180642272	1252.0	1241.111
21	West Ham				20.0	20.92	8.281102988116442	14.0	14.96	10.521723599669229	1070.0	1341.92
22	Wolves				25.0	22.086956521739133	8.680787883787812	15.0	16.173913043478262	10.547086994505261	1364.0	1452.173
23												

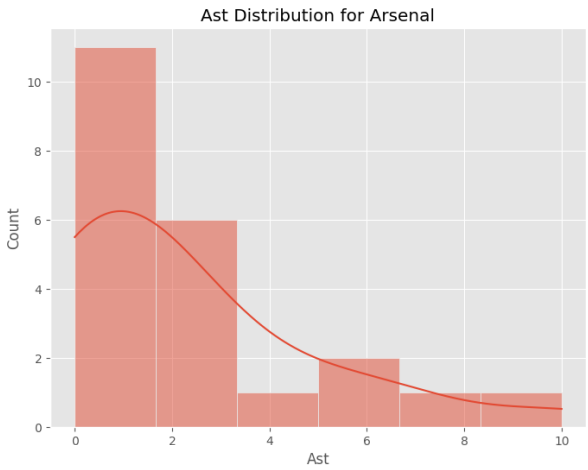
Hình 1.2: File result2.csv

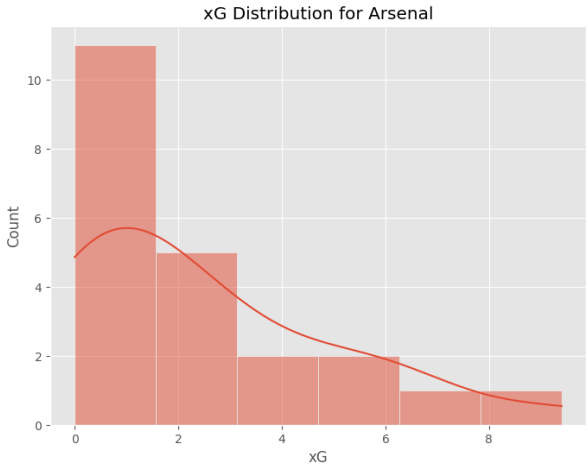
1.2.2 Yêu cầu thứ 3: Plot a histogram showing the distribution of each statistic for all players in the league and each team:

Ở đây, mã nguồn lựa chọn 3 chỉ số tấn công: Gls(Bàn thắng), Ast(Kiến tạo) và xG(Bàn thắng kì vọng) và 3 chỉ số phòng thủ: TklW(Tỉ lệ tắc bóng thành công), Lost(Đánh chặn thất bại) và Sh(Số cú sút chặn được) để plot các histogram tương ứng cho từng đội. Sau đây là kết quả mẫu của 1 đội (cụ thể là Arsenal):

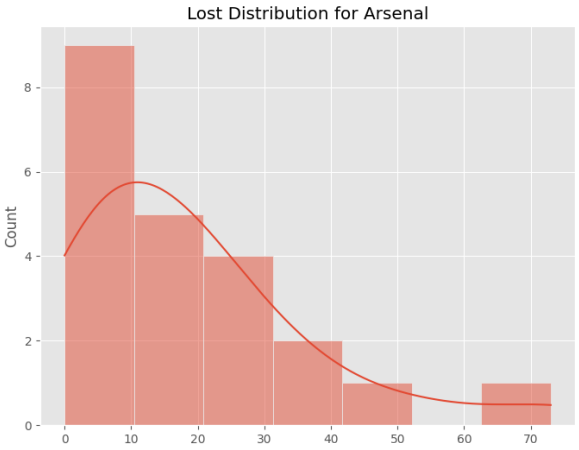
- 3 chỉ số tấn công:

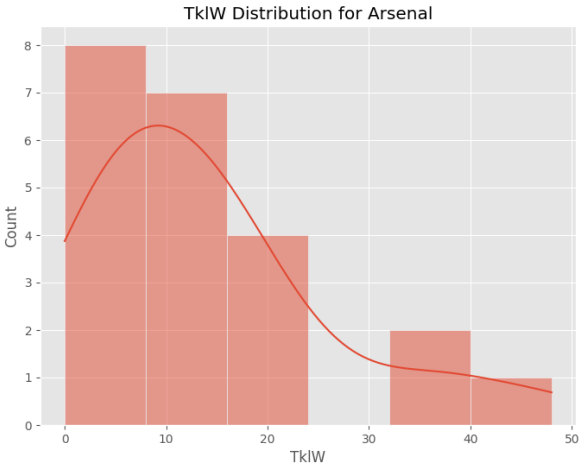


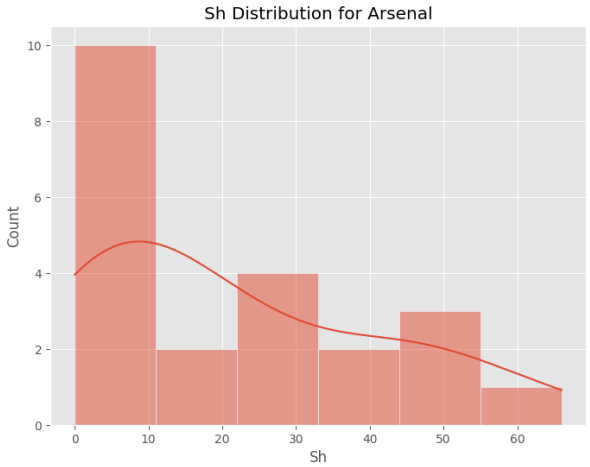




- 3 chỉ số phòng thủ:







1.2.3 Yêu cầu thứ 3: Identify the team with the highest scores for each statistic. Based on your analysis, which team do you think is performing the best in the 2024-2025 Premier League season?

Giải thích các phần quan trọng trong đoạn mã thực hiện yêu cầu:

1. **Các thư viện quan trọng**:
   - **Selenium**: Dùng để điều khiển trình duyệt tự động (truy cập trang web, tải nội dung).
   - **BeautifulSoup**: Phân tích cú pháp HTML để trích xuất dữ liệu từ trang web.
   - **pandas**: Xử lý dữ liệu dạng bảng (DataFrame).
   - **time**: Tạm dừng chương trình để đảm bảo trang web tải hoàn tất.
2. **Hàm extract_max_stats:**

```
def extract_max_stats(url, table_class):
    driver.get(url)
    time.sleep(3)
    soup = BeautifulSoup(driver.page_source, 'lxml')
    table = soup.find('table', class_=table_class)
    df = pd.read_html(str(table))[0]
    if isinstance(df.columns, pd.MultiIndex):
        df.columns = df.columns.droplevel(0)

    df = df[df['Squad'].notna()]

    results = {}
    for col in df.columns:
        if col != 'Squad' and pd.api.types.is_numeric_dtype(df[col]):
            try:
                max_row = df.loc[df[col].idxmax()]
                results[col.strip()] = (max_row['Squad'], max_row[col])
            except Exception:
                continue
    return results
```

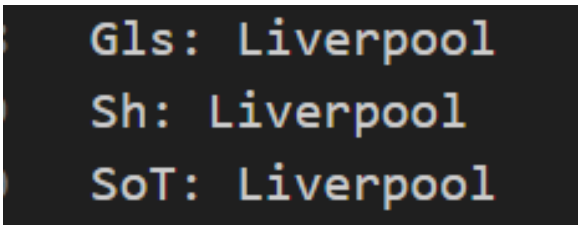
Các thao tác chính quan trọng trong hàm:

a. Tìm giá trị tối đa:

- Duyệt các cột: Với mỗi cột (trừ cột 'Squad'), kiểm tra nếu cột chứa dữ liệu số (is_numeric_dtype).
- Tìm tối đa: Sử dụng idxmax() để tìm hàng có giá trị lớn nhất trong cột, lấy tên đội (Squad) và giá trị tối đa.
- Lưu kết quả: results là một từ điển với key là tên cột, value là tuple (đội, giá trị tối đa).

```
results = {}
for col in df.columns:
    if col != 'Squad' and pd.api.types.is_numeric_dtype(df[col]):
        try:
            max_row = df.loc[df[col].idxmax()]
            results[col.strip()] = (max_row['Squad'], max_row[col])
        except Exception:
            continue
return results
```

1 phần nhỏ của file kết quả:



Dựa trên file kết quả thu được, các chỉ số tấn công quan trọng như Gls (Bàn thắng), SoT (Số cú sút trúng đích), xG(Bàn thắng kì vọng), chỉ số phòng thủ như Sh(Số cú sút chặn được),.. thì Liverpool đều là đội dẫn đầu. Bằng chứng là Liverpool dẫn đầu về số trận thắng (W)
=> Liverpool là đội thi đấu tốt nhất trong mùa giải này.

1.3 Câu 3:

1.3.1 Yêu cầu thứ 2: How many groups should the players be classified into? Why? Provide your comments on the results.

Sau khi áp dụng thuật toán phân cụm K-means lên thống kê cầu thủ từ bộ dữ liệu Premier League, số lượng cụm tối ưu (k) thường được xác định là: 4
Điều này được xác định thông qua các phương pháp như:

- **Phương pháp Elbow, trong đó tổng bình phương khoảng cách trong cụm giảm dần và bắt đầu chững lại ở khoảng k = 4.**
- **Điểm Silhouette, đạt đỉnh hoặc giữ ở mức cao tại k = 4, cho thấy các cụm được phân chia rõ ràng và có tính kết dính tốt.**

Diễn giải 4 cụm:

1. **Cụm 1: Cầu thủ tấn công**
   Cao về số bàn thắng, xG (bàn thắng kỳ vọng), số cú sút, đường chuyền quyết định và số lần chạm bóng trong vòng cấm đối phương. Thường bao gồm tiền đạo và tiền vệ tấn công.

2. **Cụm 2: Tiền vệ kiến thiết**
   Cao về số đường chuyền, bóng chọc khe, hành động tạo cơ hội dứt điểm (SCA), hành động tạo bàn thắng (GCA). Thường là tiền vệ trung tâm hoặc tiền vệ tấn công kiểm soát nhịp độ và tạo cơ hội.

3. **Cụm 3: Cầu thủ phòng ngự**
   Cao về số pha tắc bóng, cắt bóng, phá bóng và chiến thắng tranh chấp trên không. Chủ yếu là trung vệ và tiền vệ phòng ngự.

4. **Cụm 4: Thủ môn**
   Cao về số pha cứu thua, số trận giữ sạch lưới, cản phá penalty và các chỉ số riêng dành cho thủ môn. Cụm này được phân tách rõ ràng do vai trò và chỉ số đặc thù.

Giải thích mã nguồn thực hiện yêu cầu:

1. **Tải dữ liệu:**

```
df = pd.read_csv("results.csv")
```

Tải tập tin results.csv chứa dữ liệu cầu thủ vào một DataFrame dưới biến df

2. **Lọc các cột số liệu cần thiết:**

```
non_numeric = ['Player', 'Team', 'Position', 'Nation', 'Comp', 'Age']
df_numeric = df.drop(columns=[col for col in non_numeric if col in df.columns], errors='ignore')
```

Loại bỏ các cột không phải số (tên cầu thủ, đội, vị trí, v.v.) để chỉ giữ lại các thống kê số dùng cho phân cụm.

3. **Chuyển đổi dữ liệu và xử lý giá trị thiếu**

```
df_numeric = df_numeric.apply(pd.to_numeric, errors='coerce').fillna(0)
```

Chuyển mọi cột sang kiểu số. Nếu có lỗi chuyển đổi (ví dụ chứa văn bản), sẽ thay thế bằng NaN, sau đó điền 0.

4. **Chuẩn hóa dữ liệu:**

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_numeric)
```

Chuẩn hóa dữ liệu (z-score) để các thống kê có cùng thang đo, tránh ảnh hưởng đến phân cụm.

5. **Dùng Elbow Method để tìm số cụm tối ưu (k)**

```
inertias = []
k_range = range(1, 11)

for k in k_range:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    km.fit(X_scaled)
    inertias.append(km.inertia_)
```

- Thử nhiều giá trị k (từ 1 đến 10).
- Tính Inertia (tổng khoảng cách nội cụm), để dùng cho biểu đồ Elbow.

6. **Vẽ biểu đồ elbow:**

```
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertias, 'bo-')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```

Đoạn mã này dùng thư viện Matplotlib để trực quan hóa Elbow Method nhằm xác định số lượng cụm tối ưu trong phân cụm K-means:

- Tạo một biểu đồ mới với kích thước 8 (rộng) x 5 (cao) inch.
- Vẽ đường biểu diễn:
  - k_range là trục hoành (giá trị k, số cụm thử nghiệm).
  - inertias là trục tung (độ đo "inertia" – tổng bình phương khoảng cách từ mỗi điểm đến tâm cụm).
  - 'bo-' là định dạng đường:
    - 'b': màu xanh dương (blue),
    - 'o': các điểm được đánh dấu bằng vòng tròn,
    - '-': nối các điểm bằng đường thẳng.
- Gán nhãn cho trục X: "Number of Clusters (k)".
- Gán nhãn cho trục Y: "Inertia" – tổng bình phương khoảng cách trong cụm.

- Gán tiêu đề biểu đồ là "Elbow Method For Optimal k".
- Hiển thị biểu đồ.

7. **Áp dụng K-means với số cụm k đã chọn và hiển thị cầu thủ theo từng cụm:**

```
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertias, 'bo-')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```

- Gán mỗi cầu thủ vào một trong 4 cụm dựa trên dữ liệu đã chuẩn hóa.
- In danh sách tên cầu thủ thuộc mỗi cụm.

1.3.2 Yêu cầu thứ 3: Use PCA to reduce the data dimensions to 2, then plot a 2D cluster of the data points.

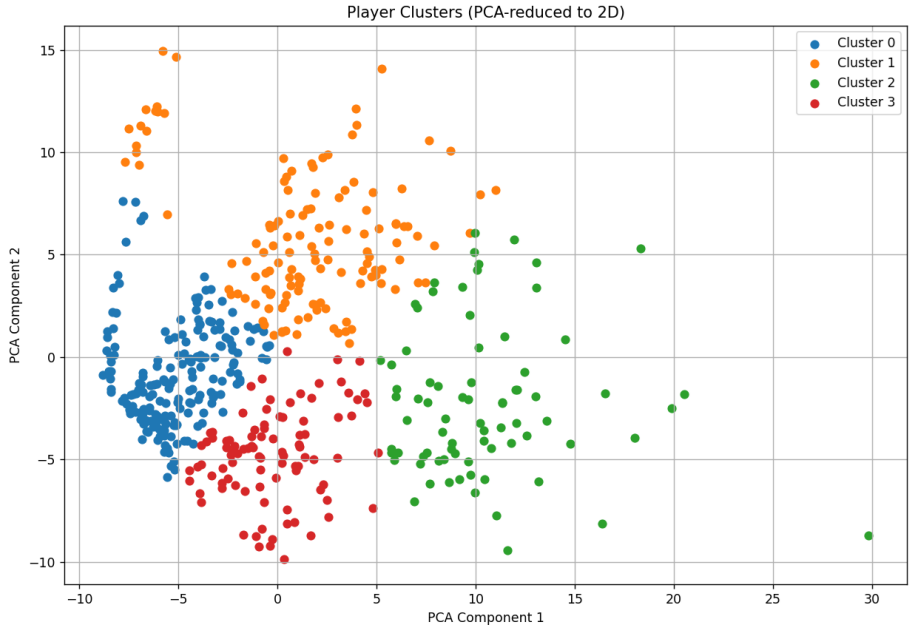
Ngoài các bước như Tải dữ liệu, Lọc các cột số liệu, Chuyển đổi dữ liệu, Chuẩn hóa dữ liệu, Tìm số cụm k tối ưu bằng phương pháp Elbow đã giải thích ở trên, yêu cầu này sử dụng thao tác **PCA** có sẵn trong thư viện **SkLearn** để giảm chiều dữ liệu xuống còn 2:

```
# Giảm chiều dữ liệu về 2D sử dụng PCA:
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

Cuối cùng, thực hiện thao tác vẽ biểu đồ biểu thị các cụm với k = 4:

```
# Vẽ biểu đồ các cụm
plt.figure(figsize=(10, 7))
for i in range(k):
    plt.scatter(
        X_pca[clusters == i, 0],
        X_pca[clusters == i, 1],
        label=f'Cluster {i}'
    )

plt.title('Player Clusters (PCA-reduced to 2D)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Kết quả thu được:



1.4 Câu 4:

1.4.1 Yêu cầu thứ nhất: Collect player transfer values for the 2024-2025 season from https://www.footballtransfers.com. Note that only collect for the players whose playing time is greater than 900 minutes

Giải thích các bước trong mã nguồn thực hiện yêu cầu bài toán:

1. **Các thư viện cần thiết:**
   - Xử lý dữ liệu: **pandas, os, re, unidecode**
   - Tự động hóa trình duyệt: **Selenium**
   - Phân tích HTML: **BeautifulSoup**
   - Chờ đợi động trong Selenium: **WebDriverWait, EC**
2. **Chuẩn hóa tên các cầu thủ:** Loại bỏ dấu, ký tự đặc biệt, viết thường tất cả, thay khoảng trắng thừa => Phục vụ so sánh tên cầu thủ:

```python
# Hàm chuẩn hóa tên cầu thủ
def normalize_name(name):
    if not isinstance(name, str):
        return ""
    name = unidecode(name.strip()).lower()
    name = re.sub(r'\s+', ' ', name)
    name = re.sub(r'[^a-z0-9\s]', '', name)
    return name
```

3. **Cài đặt Edge Webdriver:** Kiểm tra xem WebDriver có tồn tại tại đường dẫn chỉ định không. Nếu không, dừng chương trình với thông báo lỗi.

```python
# Đường dẫn đến Edge WebDriver
edge_driver_path = "C:\\Webdrivers\\msedgedriver.exe"
if not os.path.isfile(edge_driver_path):
    raise FileNotFoundError(f"WebDriver không tồn tại tại: {edge_driver_path}")
```

4. **Đọc file CSV danh sách cầu thủ:** Đọc danh sách cầu thủ đã thi đấu >900 phút. Chuẩn hóa tên và đưa vào set để so sánh nhanh hơn.

```
try:
    players_df = pd.read_csv("players_over_900_minutes.csv")
    print("Dữ liệu CSV (5 dòng đầu):")
    print(players_df.head())
    print(f"Số giá trị trống trong cột Player: {players_df['Player'].isna().sum()}")
    players_df["Player"] = players_df["Player"].astype(str).str.strip()
    player_set = set(normalize_name(player) for player in players_df["Player"] if player != "nan")
    print("Danh sách cầu thủ từ CSV (chuẩn hóa):")
    for player in sorted(player_set):
        print(player)
except Exception as e:
    raise Exception(f"Lỗi khi đọc file CSV: {e}")
```

5. **Mở trình duyệt và truy cập trang web:** Mở trình duyệt Microsoft Edge bằng Selenium và truy cập URL chứa dữ liệu chuyển nhượng cầu thủ:

```
# Cấu hình Selenium
service = Service(edge_driver_path)
options = webdriver.EdgeOptions()
driver = webdriver.Edge(service=service, options=options)
url = "https://www.footballtransfers.com/us/players/uk-premier-league"
```

6. **Tải thêm nội dung trong trang:** Cuộn xuống cuối trang nhiều lần để tải thêm dữ liệu (trang web có thể sử dụng lazy loading)

```
    for _ in range(5):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(3)
```

7. **Tìm bảng dữ liệu trong trang:**

```
table = None
tables = soup.find_all("table")
for tbl in tables:
    if tbl.find("tbody"):
        headers = [th.text.strip().lower() for th in tbl.find_all("th")]
        if any(keyword in headers for keyword in ["player", "name", "market value", "value"]):
            table = tbl
            break
```

- Tìm bảng HTML chứa thông tin cầu thủ bằng BeautifulSoup.
- Xác minh bảng dựa trên các từ khóa như "player", "name", "market value" v.v.

8. **Trích xuất thông tin từng hàng (cầu thủ)**:

```
rows = table.find("tbody").find_all("tr") if table.find("tbody") else table.find_all("tr")
for row in rows:
    cols = row.find_all("td")
    if len(cols) < 2:
        continue
    print("Các cột trong hàng:", [col.text.strip() for col in cols])
    # Thử cột đầu tiên hoặc cột thứ hai cho tên cầu thủ
    player_name = None
    for i in [0, 1]:
        if cols[i].find("a"):
            player_name = cols[i].find("a").text.strip()
            break
        elif cols[i].text.strip():
            player_name = cols[i].text.strip()
    if not player_name:
        continue
    all_players.append(player_name)
    normalized_player_name = normalize_name(player_name)
    if normalized_player_name in player_set:
        market_value = cols[-1].text.strip()
        data.append({
            "Player": player_name,
            "Market Value": market_value
        })
    else:
        continue
```

- Lấy tên cầu thủ từ cột 0 hoặc 1.
- Nếu tên khớp với danh sách trong file CSV (các cầu thủ chơi hơn 900 phút) thì lấy thêm dữ liệu "market value".
- Dữ liệu thu thập được sẽ lưu vào danh sách data.

9. **Chuyển sang trang tiếp theo nếu có:**

```
    try:
        next_button = WebDriverWait(driver, 5).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, ".pagination_next_button.btn.next-btn:not(.disabled)"))
        )
        driver.execute_script("arguments[0].click();", next_button)
        time.sleep(3)
        for _ in range(5):
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
            time.sleep(3)
    except:
        print("Không tìm thấy nút Next hoặc đã đến trang cuối.")
        break
```

10. **Tạo DataFrame, xuất ra file CSV và đóng trình duyệt:**

```
    print("Tất cả cầu thủ tìm thấy trên trang web:")
    for player in sorted(all_players):
        print(player)

    df = pd.DataFrame(data)
    if df.empty:
        print("Không tìm thấy cầu thủ nào khớp với danh sách trong file CSV.")
    else:
        print("Dữ liệu thu thập được:")
        print(df)
        df.to_csv("filtered_premier_league_players.csv", index=False, encoding="utf-8")
        print("Dữ liệu đã được lưu vào filtered_premier_league_players.csv")

finally:
    driver.quit()
```

1.4.2 Yêu cầu thứ 2: Propose a method for estimating player values. How do you select feature and model?

Đầu tiên, ta cần gộp giá trị chuyển nhượng thu được từ yêu cầu 1 vào file result.csv, thu được file final_result.csv. => Chọn Linear Regression.

Để sử dụng Linear Regression nhằm ước tính giá trị cầu thủ (Market Value) dựa trên file dữ liệu final_result.csv, chúng ta cần thực hiện các bước sau. File này chứa nhiều đặc trưng (features) liên quan đến hiệu suất thi đấu của các cầu thủ, và cột Market Value là biến mục tiêu (target variable). Dưới đây là giải thích chi tiết về cách chọn đặc trưng và xây dựng mô hình Linear Regression:

1. **Hiểu dữ liệu và mục tiêu:**
    - Mục tiêu: Dự đoán giá trị thị trường của cầu thủ (Market Value) dựa trên các đặc trưng trong dữ liệu.
    - Biến mục tiêu: Market Value (được biểu diễn dưới dạng giá trị tiền, ví dụ: €18.7M).
    - Đặc trưng (Features): Các cột khác trong file như tuổi (Age), số phút thi đấu (Min), số bàn thắng (Gls), số kiến tạo (Ast), số thẻ vàng (CrdY), số đường chuyền hoàn thành (Cmp), v.v.
2. **Chuẩn bị dữ liệu:** Trước khi áp dụng Linear Regression, cần xử lý dữ liệu để đảm bảo phù hợp với mô hình:
    a. **Làm sạch dữ liệu:**
        - Xử lý giá trị thiếu: Kiểm tra các cột có giá trị thiếu (NaN hoặc N/a). Ví dụ, một số cột như GA, SoTA chỉ áp dụng cho thủ môn, nên có thể bỏ qua cho các vị trí khác hoặc điền giá trị 0 nếu phù hợp.
        - Chuyển đổi định dạng:
            – Cột Market Value cần được chuyển từ dạng chuỗi (ví dụ: "€18.7M") sang số thực (18.7 triệu euro). Có thể dùng hàm để loại bỏ ký tự "€" và "M" rồi nhân với 1,000,000.
            – Cột Age có định dạng "26-353" (tuổi và số ngày). Chỉ lấy phần tuổi (26) để đơn giản hóa.
        - Loại bỏ cột không liên quan: Các cột như Player, Nation, Team, Matches không phải là số và có thể không trực tiếp liên quan đến giá trị thị trường, nên có thể bỏ qua hoặc mã hóa nếu cần (ví dụ: mã hóa one-hot cho Position).

    b. **Chuyển đổi biến phân loại:**
        - Cột Position (ví dụ: GK, DF, MF, FW) là biến phân loại. Có thể mã hóa bằng One-Hot Encoding để chuyển thành các cột nhị phân (0 hoặc 1) cho mỗi vị trí.
        - Các cột như Nation hoặc Team có thể được mã hóa tương tự, nhưng vì số lượng giá trị riêng biệt lớn, có thể bỏ qua để tránh làm tăng chiều dữ liệu quá nhiều.

c. **Chuẩn hóa dữ liệu:** Linear Regression nhạy cảm với quy mô của các đặc trưng. Các cột như Min (phút thi đấu) có giá trị lớn (hàng nghìn), trong khi Gls (bàn thắng) thường nhỏ hơn. Sử dụng StandardScaler hoặc MinMaxScaler để chuẩn hóa các đặc trưng về cùng thang đo (ví dụ: trung bình 0, độ lệch chuẩn 1).

d. **Xử lý outlier:** Kiểm tra các giá trị bất thường (outlier) trong các cột như Market Value, Gls, hoặc Min. Ví dụ, một cầu thủ có giá trị thị trường cực cao (như €120.3M của Alexander Isak) có thể làm mô hình bị lệch. Có thể áp dụng log transform cho Market Value để giảm độ lệch (skewness) hoặc loại bỏ outlier nếu cần.

3. **Chọn đặc trưng (Feature Selection):**
Việc chọn đặc trưng là bước quan trọng để đảm bảo mô hình Linear Regression hoạt động hiệu quả. Dựa trên dữ liệu trong file, các đặc trưng có thể được chọn dựa trên:

a. **Hiểu ngữ cảnh bóng đá:**

- Tuổi (Age): Tuổi thường ảnh hưởng mạnh đến giá trị cầu thủ. Cầu thủ trẻ (20-25 tuổi) thường có giá trị cao hơn do tiềm năng phát triển, trong khi cầu thủ lớn tuổi (>30) có thể giảm giá trị.
- Hiệu suất thi đấu:
  - Gls (bàn thắng), Ast (kiến tạo), G+A (bàn thắng + kiến tạo): Quan trọng với các vị trí tấn công (FW, MF).
  - xG, xAG (expected goals, expected assists): Đo lường khả năng đóng góp dự kiến, phản ánh chất lượng cơ hội.
  - Cmp, Cmp% (số đường chuyền hoàn thành, tỷ lệ chuyền chính xác): Quan trọng với tiền vệ và hậu vệ.
  - PrgC, PrgP, PrgR (Progressive Carries, Passes, Receives): Đo lường khả năng đưa bóng lên phía trước.
  - SCA, GCA (Shot-Creating Actions, Goal-Creating Actions): Phản ánh khả năng tạo cơ hội.
- Thời gian thi đấu: Min (phút thi đấu), Starts (số trận đá chính): Cầu thủ thi đấu nhiều thường có giá trị cao hơn.
- Kỷ luật: CrdY, CrdR (thẻ vàng, thẻ đỏ): Có thể ảnh hưởng tiêu cực đến giá trị.
- Thống kê thủ môn: GA (bàn thua), Saves, Save%, CS (sạch lưới): Quan trọng để đánh giá thủ môn.
- Vị trí (Position): Giá trị cầu thủ phụ thuộc vào vai trò (thủ môn, hậu vệ, tiền vệ, tiền đạo). Mã hóa one-hot để đưa vào mô hình.

b. **Phân tích thống kê:**

- Tương quan (Correlation): Tính ma trận tương quan giữa các đặc trưng và Market Value. Các đặc trưng có tương quan cao (ví dụ: Gls, xG, Min) nên được ưu tiên. Ví dụ: Gls và xG có thể tương quan cao với Market Value cho tiền đạo.

- Kiểm tra đa cộng tuyến (Multicollinearity): Các đặc trưng như Gls và xG, hoặc Cmp và Att (số đường chuyền hoàn thành và tổng số đường chuyền) có thể tương quan mạnh với nhau. Sử dụng Variance Inflation Factor (VIF) để kiểm tra và loại bỏ các đặc trưng có VIF > 10.

c. Phương pháp tự động:

- Recursive Feature Elimination (RFE): Sử dụng RFE với Linear Regression để chọn ra các đặc trưng quan trọng nhất.
- Lasso Regression: Áp dụng Lasso (L1 regularization) để tự động loại bỏ các đặc trưng không quan trọng bằng cách đưa hệ số của chúng về 0.

d. Đặc trưng được chọn (ví dụ): Dựa trên file, các đặc trưng tiềm năng bao gồm:

- Age, Min, Gls, Ast, xG, xAG, Cmp, Cmp%, PrgC, PrgP, PrgR, SCA, GCA, CrdY, CrdR.
- Cột one-hot từ Position (GK, DF, MF, FW).
- Đối với thủ môn, có thể xây mô hình riêng với các đặc trưng như Saves, Save%, CS.

4. **Xây dựng mô hình Linear Regression:**

a. Các bước xây dựng mô hình:

i. **Chia dữ liệu:**

- Chia dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test) với tỷ lệ 80:20 hoặc 70:30.
- Sử dụng train_test_split từ scikit-learn.

ii. **Huấn luyện mô hình:**

- Sử dụng LinearRegression từ scikit-learn để fit mô hình trên tập train.
- Đoạn mã cụ thể:

```
# Chọn các đặc trưng số để làm biến độc lập
features = numeric_columns.drop(['Market Value', 'Rk'])  # Loại bỏ Market Value và Rk
X = data[features]
y = data['Market Value']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Khởi tạo và huấn luyện mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)
```

iii. **Dự đoán và đánh giá:**

- Dự đoán trên tập test: y_pred = model.predict(X_test_scaled).
- Đánh giá mô hình bằng các chỉ số:
  - Mean Squared Error (MSE): Đo lường sai số bình phương trung bình.
  - R-squared (R^2): Đo lường mức độ giải thích của mô hình ($0 \leq R \leq 1$).
  - Đoạn mã cụ thể:

```
# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
print(f'R² Score: {r2:.2f}')
```

b. Kiểm tra giả định của Linear Regression:

- Tính tuyến tính: Kiểm tra scatter plot giữa các đặc trưng và Market Value. Nếu không tuyến tính, có thể thử log transform hoặc các mô hình phi tuyến (như Random Forest).
- Độc lập: Các quan sát (cầu thủ) được giả định độc lập.
- Phân phối chuẩn của sai số: Kiểm tra histogram của sai số (y_test - y_pred).
- Đồng nhất phương sai (Homoscedasticity): Vẽ scatter plot của sai số so với giá trị dự đoán để kiểm tra.

5. **Ví dụ minh họa:**
   Giả sử chúng ta chọn các đặc trưng: Age, Min, Gls, Ast, xG, xAG, Cmp%, PrgC, PrgP, Position. Quy trình sẽ như sau:

   (a) Làm sạch và chuẩn hóa dữ liệu.

   (b) Mã hóa Position thành one-hot.

   (c) Chuẩn hóa các đặc trưng số bằng StandardScaler.

   (d) Chia dữ liệu train/test, huấn luyện mô hình Linear Regression.

   (e) Đánh giá R^2 và MSE, kiểm tra hệ số để xem đặc trưng nào quan trọng (ví dụ: Gls có hệ số cao cho tiền đạo).

   (f) Tinh chỉnh bằng Ridge/Lasso hoặc thử Random Forest nếu R^2 thấp.

Tóm lại, Linear Regression là một cách tiếp cận đơn giản nhưng hiệu quả để dự đoán giá trị cầu thủ nếu dữ liệu được xử lý tốt và các đặc trưng được chọn cẩn thận. Tuy nhiên, để đạt độ chính xác cao hơn, có thể cần kết hợp với các mô hình phức tạp hơn hoặc thêm dữ liệu bổ sung.