HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH NGÔN NGỮ PYTHON

BÁO CÁO BÀI TẬP LỚN

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**

Lớp : **D23CQCE06-B**

Họ Tên : **NGUYỄN PHẠM ANH PHÚC**

MSV : **B23DCCE078**

Hà Nội – 2025

MỤC LỤC

LỜI MỞ ĐẦU

Em thực hiện báo cáo này nhằm hoàn thành Bài tập lớn số 1 của học phần Lập trình Python. Mục tiêu chính của bài tập là áp dụng các kiến thức về lập trình và xử lý dữ liệu để xây dựng một hệ thống thu thập, phân tích và trực quan hóa dữ liệu thống kê của các cầu thủ thi đấu tại giải **Premier League mùa giải 2024–2025**.

Cụ thể, em đã phát triển một chương trình Python có khả năng tự động thu thập dữ liệu từ trang FBref đối với các cầu thủ có thời gian thi đấu trên 90 phút. Dữ liệu thu được được xử lý, lưu trữ và phân tích để rút ra các thống kê quan trọng. Em cũng đã sử dụng thuật toán K-means để phân cụm cầu thủ dựa trên các đặc điểm chuyên môn, đồng thời áp dụng PCA để trực quan hóa kết quả phân cụm. Bên cạnh đó, em còn thu thập dữ liệu giá trị chuyển nhượng từ Football Transfers và đề xuất một phương pháp ước lượng giá trị cầu thủ dựa trên mô hình học máy.

Báo cáo này trình bày chi tiết quá trình thực hiện, kết quả đạt được, các biểu đồ trực quan hóa, cũng như những nhận xét, đánh giá cá nhân trong suốt quá trình làm bài.

I Bài Tập 1

1. Cách triển khai

Bước 1: Thiết lập

```
# Danh sách bảng cần cào
table_links = {
    'Standard Stats': ('https://fbref.com/en/comps/9/stats/Premier-League-Stats', 'stats_standard'),
    'Shooting': ('https://fbref.com/en/comps/9/shooting/Premier-League-Stats', 'stats_shooting'),
    'Passing': ('https://fbref.com/en/comps/9/passing/Premier-League-Stats', 'stats_passing'),
    'Goal and Shot Creation': ('https://fbref.com/en/comps/9/gca/Premier-League-Stats', 'stats_gca'),
    'Defense': ('https://fbref.com/en/comps/9/defense/Premier-League-Stats', 'stats_defense'),
    'Possession': ('https://fbref.com/en/comps/9/possession/Premier-League-Stats', 'stats_possession'),
    'Miscellaneous': ('https://fbref.com/en/comps/9/misc/Premier-League-Stats', 'stats_misc'),
    'Goalkeeping': ('https://fbref.com/en/comps/9/keepers/Premier-League-Stats', 'stats_keeper')
}

# Khởi tạo WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
```

- Tạo danh sách các bảng thống kê cần cào từ FBref (URL và ID bảng).
- Khởi động trình duyệt giả lập bằng Selenium.

Bước 2: Duyệt qua từng bảng thống kê

* Với mỗi bảng:

```
for name, (url, table_id) in table_links.items():
    print(f'\nCào bảng: {name}')
    driver.get(url)
```

- Truy cập URL tương ứng.

```
try:
    WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, table_id)))
except Exception as e:
    print(f'Lỗi load bảng {name}: {e}')
    continue
```

- Chờ bảng thống kê hiện ra. Nếu bảng bị lỗi thì bỏ qua.

```
soup = BeautifulSoup(driver.page_source, 'html.parser')
table = soup.find('table', id=table_id)
```

- Sử dụng BeautifulSoup để trích xuất bảng HTML bằng cách tìm các thẻ ‘id’ của bảng

```
df = pd.read_html(StringIO(str(table)))[0]
```

- Đọc bảng vào pandas DataFrame.

```
if isinstance(df.columns, pd.MultiIndex):
    new_cols = []
    for col in df.columns:
        group = col[0].strip() if col[0].strip() and 'Unnamed' not in col[0] else ''
        subgroup = col[1].strip() if col[1].strip() else col[0].strip()
        col_name = f"{group} {subgroup}" if group and group != subgroup else subgroup
        new_cols.append(col_name.strip())
    df.columns = new_cols
else:
    df.columns = [col.strip() for col in df.columns]
```

- Chuẩn hóa tên cột (nếu là MultiIndex thì gộp lại):

+ **df.columns** là thuộc tính chứa tên các cột của DataFrame.

+ **if isinstance(df.columns, pd.MultiIndex)** kiểm tra xem các cột có phải là MultiIndex hay không (tức là có nhiều cấp độ, như "Group" và "Subgroup" trong tên cột).

+ **group = col[0].strip():** Lấy phần đầu tiên của tên cột (thường là "group").

+ **subgroup = col[1].strip() if col[1].strip() else col[0].strip():** Lấy phần thứ hai (subgroup) nếu có, nếu không thì gán giá trị của group cho nó.

+ **col_name = f"{group} {subgroup}" if group and group != subgroup else subgroup**: Tạo tên cột mới từ group và subgroup. Nếu chúng giống nhau thì chỉ lấy subgroup.

+ **new_cols.append(col_name.strip()):** Thêm tên cột đã làm sạch vào danh sách new_cols.

+ Sau đó, danh sách mới **new_cols** được gán lại cho **df.columns**, thay thế các tên cột cũ.

```
player_col = next((col for col in df.columns if 'player' in col.lower() and 'rk' not in col.lower()), None)
if not player_col:
    print(f"Lỗi: Không tìm thấy cột 'Player' trong {name}")
    continue
```

- Tìm và đổi tên cột chứa tên cầu thủ thành Player.

```
df = df.loc[df[player_col].notna() & (df[player_col] != player_col)].drop_duplicates(subset=player_col)
df = df.rename(columns={player_col: 'Player'})
```

- Làm sạch bảng

```
if name == 'Passing':
    df = df.rename(columns={'1/3': 'Passing 1/3'})
if name == 'Standard Stats':
    required_cols = ['Player', 'Nation', 'Squad', 'Pos', 'Age', 'Playing Time Min', 'Playing Time MP' , 'Playing Time Starts',
                     'Performance Gls', 'Performance Ast', 'Performance CrdY', 'Performance CrdR',
                     'Expected xG',  'Expected xAG','Progression PrgC', 'Progression PrgP', 'Progression PrgR',
                     'Per 90 Minutes Gls', 'Per 90 Minutes Ast', 'Per 90 Minutes xG', 'Per 90 Minutes xAG']
elif name == 'Shooting':
    required_cols = ['Player', 'Standard SoT%', 'Standard SoT/90', 'Standard G/Sh', 'Standard Dist']
elif name == 'Passing':
    required_cols = ['Player', 'Total Cmp', 'Total Cmp%', 'Total TotDist', 'Short Cmp', 'Medium Cmp', 'Long Cmp',
                     'KP', 'Passing 1/3', 'PPA', 'CrsPA', 'PrgP']
elif name == 'Goal and Shot Creation':
    required_cols = ['Player', 'SCA', 'SCA SCA90', 'GCA', 'GCA GCA90']
elif name == 'Defense':
    required_cols = ['Player', 'Tackles Tkl', 'Tackles TklW', 'Challenges Att', 'Challenges Lost',
                     'Blocks', 'Blocks Sh', 'Blocks Pass', 'Int']
elif name == 'Possession':
    required_cols = ['Player', 'Touches', 'Touches Def Pen', 'Touches Def 3rd', 'Touches Mid 3rd', 'Touches Att 3rd', 'Touches Att Pen',
                     'Take-Ons Att', 'Take-Ons Succ%', 'Take-Ons Tkld%',
                     'Carries', 'Carries PrgDist', 'Carries PrgC', 'Carries 1/3', 'Carries CPA', 'Carries Mis', 'Carries Dis',
                     'Receiving Rec', 'Receiving PrgR']
elif name == 'Miscellaneous':
    required_cols = ['Player', 'Performance Fls', 'Performance Fld', 'Performance Off', 'Performance Crs', 'Performance Recov' , 'Aerial Du
elif name == 'Goalkeeping':
    required_cols = ['Player', 'Performance GA90', 'Performance Save%', 'Performance CS%', 'Penalty Kicks Save%']
else:
    required_cols = ['Player']
```

- Lựa chọn các cột cần thiết tùy vào từng bảng.

Bước 3: Xử lý dữ liệu

```
if name == 'Goalkeeping':
    goalkeeping_df = df.copy()
else:
    if merged_df.empty:
        merged_df = df
    else:
        df = df.drop(columns=[col for col in df.columns if col in merged_df.columns and col != 'Player'])
        merged_df = pd.merge(merged_df, df, on='Player', how='outer')
```

- Nếu bảng là "Goalkeeping":

    - Lưu riêng ra **goalkeeping_df**. (Vì thủ môn là một vị trí đặc thù với hệ thống chỉ số riêng, nên cần tách dữ liệu của họ thành một bảng riêng trước khi ghép lại với các vị trí khác)

- Ngược lại:

    - Nếu bảng đầu tiên, gán làm **merged_df**.
    - Nếu không phải bảng đầu tiên, hợp nhất với **merged_df** theo cột Player.

Bước 4: Lọc cầu thủ chơi trên 90 phút

```
min_col = next((col for col in merged_df.columns if 'playing time min' in col.lower()), None)
if min_col:
    merged_df = merged_df[merged_df[min_col].notna()]
    try:
        merged_df[min_col] = merged_df[min_col].str.replace(',', '', regex=False).astype(float)
        merged_df = merged_df[merged_df[min_col] > 90]
        print(f"Lọc được {len(merged_df)} cầu thủ chơi >90 phút")
    except Exception as e:
        print(f"Lỗi lọc 'Playing Time Min': {e}")
else:
    print("Lỗi: Không tìm thấy cột 'Playing Time Min'")
```

- Tìm cột "Playing Time Min".
- Chuyển thành số, lọc các cầu thủ có số phút > 90.

Bước 5: Xử lý dữ liệu thủ môn

```python
if not goalkeeping_df.empty and not merged_df.empty:
    goalkeeping_df = goalkeeping_df[goalkeeping_df['Player'].isin(merged_df['Player'])]
    pos_col = next((col for col in merged_df.columns if 'pos' in col.lower()), None)
    if pos_col:
        goalkeeping_df['Pos'] = goalkeeping_df['Player'].map(merged_df.set_index('Player')[pos_col])
        for col in goalkeeping_df.columns:
            if col not in ['Player', 'Pos']:
                goalkeeping_df[col] = goalkeeping_df.apply(
                    lambda row: row[col] if pd.notna(row['Pos']) and 'GK' in row['Pos'] else 'N/A', axis=1
                )
        goalkeeping_df.drop(columns='Pos', inplace=True)
        merged_df = pd.merge(merged_df, goalkeeping_df, on='Player', how='left')
    else:
        print("Lỗi: Không tìm thấy cột 'Pos'")
```

- Chỉ giữ thủ môn có trong **merged_df.**
- Xác định vị trí cầu thủ từ **merged_df**.
- Với mỗi cột thống kê thủ môn:
    + Nếu không phải thủ môn, gán 'N/a’
- Hợp nhất bảng thủ môn vào **merged_df**.

Bước 6: Chọn cột mong muốn

```python
required_columns = [
    'Player', 'Nation', 'Squad', 'Pos', 'Age', 'Playing Time Min', 'Playing Time MP' , 'Playing Time Starts',
                    'Performance Gls', 'Performance Ast', 'Performance CrdY', 'Performance CrdR',
                    'Expected xG',  'Expected xAG','Progression PrgC', 'Progression PrgP', 'Progression PrgR',
                    'Per 90 Minutes Gls', 'Per 90 Minutes Ast', 'Per 90 Minutes xG', 'Per 90 Minutes xAG',
    'Performance GA90', 'Performance Save%', 'Performance CS%', 'Penalty Kicks Save%',
    'Standard SoT%', 'Standard SoT/90', 'Standard G/Sh', 'Standard Dist',
    'Total Cmp', 'Total Cmp%', 'Total TotDist', 'Short Cmp', 'Medium Cmp', 'Long Cmp',
                    'KP', 'Passing 1/3', 'PPA', 'CrsPA', 'PrgP',
    'SCA', 'SCA SCA90', 'GCA', 'GCA GCA90',
    'Tackles Tkl', 'Tackles TklW', 'Challenges Att', 'Challenges Lost',
    'Blocks', 'Blocks Sh', 'Blocks Pass', 'Int',
    'Touches', 'Touches Def Pen', 'Touches Def 3rd', 'Touches Mid 3rd', 'Touches Att 3rd', 'Touches Att Pen',
    'Take-Ons Att', 'Take-Ons Succ%', 'Take-Ons Tkld%',
    'Carries', 'Carries PrgDist', 'Carries PrgC', 'Carries 1/3', 'Carries CPA', 'Carries Mis', 'Carries Dis',
    'Receiving Rec', 'Receiving PrgR',
    'Performance Fls', 'Performance Fld', 'Performance Off', 'Performance Crs', 'Performance Recov', 'Aerial Duels Won', 'Aerial Duels Lost', 'Aerial Duels Won%'
]
```

- Xác định danh sách cột mong muốn (required_columns).
- Kiểm tra xem cột nào thực sự tồn tại trong merged_df

```
# Lọc cột có sẵn
available_columns = [col for col in required_columns if col in merged_df.columns]
if not available_columns:
    print("Lỗi: Không tìm thấy cột nào trong danh sách mong muốn")
    print(f"Các cột hiện có trong merged_df: {list(merged_df.columns)}")
else:
    # Đảm bảo Nation, Squad, Age, Pos luôn được ưu tiên
    must_have = ['Player', 'Nation', 'Squad', 'Pos', 'Age']
    final_columns = must_have + [col for col in available_columns if col not in must_have]
    final_columns = [col for col in final_columns if col in merged_df.columns]
    merged_df = merged_df[final_columns]
    print(f"Cột cuối cùng: {final_columns}")
merged_df.fillna('N/a', inplace=True)
```

- Ưu tiên giữ các cột: Player, Nation, Squad, Pos, Age.(Lý do cần ưu tiên giữ lại các cột này là vì nếu không, chương trình sẽ chỉ thu thập các chỉ số chuyên môn mà bỏ sót những thông tin cốt lõi về cầu thủ — như quốc tịch, độ tuổi, đội bóng hay vị trí thi đấu — vốn rất quan trọng để phân tích toàn diện.)

Bước 7: Xuất file

```
output_file = 'results.csv'
try:
    merged_df.to_csv(output_file, index=False, encoding='utf-8-sig')
    print(f'Xuất file {output_file} thành công!')
except Exception as e:
    output_file = 'results_backup.csv'
    print(f'Lỗi ghi file results.csv: {e}, thử lưu vào {output_file}')
    merged_df.to_csv(output_file, index=False, encoding='utf-8-sig')
    print(f'Xuất file {output_file} thành công!')
```

- Ghi **merged_df** ra file results.csv.
- Nếu lỗi, ghi ra file dự phòng results_backup.csv

2. Kết quả

Player	Nation	Squad	Pos	Age	Playing Tin	Playing Tin	Playing Tin	Performan	Performan	Performan	Performan	Expected ›	Expected ›	Progressio	Progressio	Progressio	Per 90 Min	Per 90 Min	Per 90 Min	Per 90 Min	Performan	Performan	Pe
Aaron Cre	eng ENG	West Ham	DF	35-133	589	14	7	0	0	2	0	0.1	1.1	4	24	2	0	0	0.02	0.17	N/a	N/a	N/
Aaron Ran	eng ENG	Southamp	GK	26-348	2340	26	26	0	0	2	0	0	0	0	0	0	0	0	0	0	2.31	67.5	
Aaron Wa	eng ENG	West Ham	DF	27-152	2794	32	31	2	2	1	0	1.2	2.9	98	125	139	0.06	0.06	0.04	0.09	N/a	N/a	N/
Abdoulaye	ml MLI	Everton	MF	32-116	2425	30	29	3	1	5	1	3.9	2.3	40	78	91	0.11	0.04	0.14	0.09	N/a	N/a	N/
Abdukodir	uz UZB	Manchest	DF	21-057	503	6	6	0	0	1	0	0	0.1	1	25	2	0	0	0	0.02	N/a	N/a	N/
Abdul Fata	gh GHA	Leicester	FW	21-050	579	11	6	0	2	0	0	0.4	1.6	42	17	60	0	0.31	0.06	0.24	N/a	N/a	N/
Adam Arm	eng ENG	Southamp	FW,MF	28-076	1248	20	15	2	2	4	0	3.3	1.2	25	21	79	0.14	0.14	0.24	0.09	N/a	N/a	N/
Adam Lalli	eng ENG	Southamp	MF	36-352	361	14	5	0	2	4	0	0.2	0.9	6	24	10	0	0.5	0.04	0.23	N/a	N/a	N/
Adam Smit	eng ENG	Bournemo	DF	33-363	1319	21	16	0	0	5	0	0.7	0.2	12	39	31	0	0	0.04	0.02	N/a	N/a	N/
Adam Wel	eng ENG	Brighton	DF	30-113	617	11	8	0	0	0	0	0	0.5	7	40	2	0	0	0	0.07	N/a	N/a	N/
Adam Wha	eng ENG	Crystal Pal	MF	20-329	1258	19	15	0	2	2	0	0.3	3	14	105	10	0	0.14	0.02	0.21	N/a	N/a	N/
Adama Tra	es ESP	Fulham	FW,MF	29-092	1568	32	16	2	6	2	0	3.8	4.7	87	61	143	0.11	0.34	0.22	0.27	N/a	N/a	N/
Albert Grø	dk DEN	Southamp	FW,MF	23-339	143	4	2	0	0	0	0	0.1	0	1	1	3	0	0	0.07	0.01	N/a	N/a	N/
Alejandro	ar ARG	Manchest	MF,FW	20-300	1966	32	21	5	1	2	0	6.2	3.5	130	50	260	0.23	0.05	0.28	0.16	N/a	N/a	N/
Alex Iwobi	ng NGA	Fulham	FW,MF	28-359	2721	34	32	9	6	1	0	4.5	6.5	132	196	221	0.3	0.2	0.15	0.22	N/a	N/a	N/
Alex McCa	eng ENG	Southamp	GK	35-145	450	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	2.6	70.3	
Alex Palm	eng ENG	Ipswich Tc	GK	28-260	900	10	10	0	0	2	0	0	0	0	1	0	0	0	0	0	2.5	60	
Alex Scott	eng ENG	Bournemo	MF	21-249	612	16	6	0	0	2	0	0.7	0.8	12	48	31	0	0	0.1	0.12	N/a	N/a	N/
Alexander	se SWE	Newcastle	FW	25-218	2487	31	31	22	6	1	0	19	4	77	77	196	0.8	0.22	0.69	0.15	N/a	N/a	N/
Alexis Mac	ar ARG	Liverpool	MF	26-124	2471	32	29	4	4	6	0	2.7	4.4	32	169	75	0.15	0.15	0.1	0.16	N/a	N/a	N/
Ali Al Ham	iq IRQ	Ipswich Tc	FW	23-057	134	11	0	0	0	3	0	0.4	0.1	4	3	14	0	0	0.24	0.05	N/a	N/a	N/
Alisson	br BRA	Liverpool	GK	32-207	2058	23	23	0	0	0	0	0	0.6	0	0	0	0	0	0	0.03	0.83	72.1	
Alphonse /	fr FRA	West Ham	GK	32-059	1990	23	22	0	0	0	0	0	0	0	0	0	0	0	0	0	1.72	63.1	
Amad Diall	ci CIV	Manchest	FW,MF	22-290	1594	22	17	6	6	3	0	4.1	3.9	92	53	159	0.34	0.34	0.23	0.22	N/a	N/a	N/
Amadou O	be BEL	Aston Villa	MF	23-254	1348	22	17	3	0	4	0	2.1	0.3	20	58	13	0.2	0	0.14	0.02	N/a	N/a	N/
Andreas Pe	br BRA	Fulham	MF	29-116	1879	31	23	2	4	7	0	3.5	4.5	27	99	78	0.1	0.19	0.17	0.21	N/a	N/a	N/
Andrew Rc	sct SCO	Liverpool	DF	31-047	2218	30	26	0	0	3	1	0.9	4.1	57	163	90	0	0	0.04	0.17	N/a	N/a	N/
André	br BRA	Wolves	MF	23-285	2170	29	27	0	0	7	0	0.4	0.7	8	92	11	0	0	0.01	0.03	N/a	N/a	N/
André Ona	cm CMR	Manchest	GK	29-025	2880	32	32	0	0	0	0	0	0.2	0	1	0	0	0	0	0.01	1.31	69.5	
Andrés Ga	es ESP	Aston Villa	DF,MF	22-079	318	7	5	0	0	0	0	0	0.2	26	15	28	0	0	0.01	0.05	N/a	N/a	N/

Hình 1.1: Một phần của bảng thống kê cầu thủ

II Bài Tập 2

* Tính toán 3 người chơi có điểm cao nhất và thấp nhất

1. Cách triển khai

- Bước 1: Chọn 3 chỉ số tấn công và 3 chỉ số phòng ngự

```
attacking_cols = ['Standard SoT/90', 'Standard G/Sh', 'Standard Dist']
defensive_cols = ['Tackles Tkl', 'Tackles TklW', 'Blocks']
```

- Bước 2: Đọc dữ liệu

```
try:
    df = pd.read_csv(CSV_FILE, encoding=ENCODING)
    print(f"Đã đọc file '{CSV_FILE}' thành công.")
except Exception as e:
    print(f"Lỗi khi đọc file '{CSV_FILE}': {e}")
    exit()
```

- Đọc file results.csv bằng pandas, với encoding là 'utf-8-sig'.

- Nếu có lỗi khi đọc thì thoát chương trình.

- Bước 3: Chuyển đổi dữ liệu thành chuỗi nếu cần (Các chỉ số có kí hiệu ‘%’ có thể bị bỏ qua nên cần chuyển đổi)

```
for col in df.columns:
    if df[col].dtype == 'object' and col not in ['Player', 'Nation', 'Squad', 'Pos', 'Age']:
        if df[col].str.contains('%', na=False).any():
            df[col] = df[col].str.rstrip('%').astype(float)
        else:
            df[col] = pd.to_numeric(df[col], errors='coerce')
```

- Với các cột có kiểu dữ liệu object (chuỗi) và không phải là Player, Nation, Squad, Pos, Age:
  - Nếu có % thì xóa dấu % và chuyển thành float.
  - Nếu không có %, cố gắng chuyển đổi sang số (float/int).

- Bước 4: Lọc các cột số từ danh sách chọn lọc

```
selected_columns = attacking_cols + defensive_cols
numeric_columns = [col for col in selected_columns if col in df.columns and pd.api.types.is_numeric_dtype(df[col])]

if not numeric_columns:
    print("Không tìm thấy cột số hợp lệ.")
    exit()

print("Sử dụng các cột sau:")
for i, col in enumerate(numeric_columns, 1):
    print(f"{i}. {col}")
```

- Tạo danh sách các cột được chọn (attacking_cols + defensive_cols).

- Giữ lại các cột tồn tại trong DataFrame và có kiểu số (numeric_columns).

- Nếu không có cột nào hợp lệ, thoát chương trình.

- Bước 5: Ghi bottom/top 3 cầu thủ

```
with open(TOP_PLAYERS_FILE, 'w', encoding=ENCODING) as f:
    for col in numeric_columns:
        f.write(f"Statistic: {col}\n")
        f.write("=" * (len(col) + 11) + "\n")

        # Top 3
        f.write("Top 3 cầu thủ:\n")
        top_3 = df[['Player', col]].dropna().sort_values(by=col, ascending=False).head(TOP_N)
        f.write(top_3.to_string(index=False) + "\n\n")

        # Bottom 3
        f.write("Bottom 3 cầu thủ:\n")
        bottom_3 = df[['Player', col]].dropna().sort_values(by=col, ascending=True).head(TOP_N)
        f.write(bottom_3.to_string(index=False) + "\n")
        f.write("-" * 50 + "\n\n")
print(f"Lưu file top/bottom 3 vào '{TOP_PLAYERS_FILE}'")
```

- Mở file top_3.txt để ghi.
- Với mỗi chỉ số trong numeric_columns:
  - Lấy top 3 cầu thủ theo giá trị giảm dần.
  - Lấy bottom 3 cầu thủ theo giá trị tăng dần.
  - Ghi vào file theo định dạng rõ ràng.

2. Kết quả

- Dưới đây là toàn bộ kết quả:

```
Statistic: Standard SoT/90
==========================
Top 3 players:
         Player  Standard SoT/90
  Donyell Malen             2.64
Luis Sinisterra             2.29
    Jáder Durán             2.12

Bottom 3 players:
         Player  Standard SoT/90
Aaron Cresswell              0.0
 Arijanet Muric              0.0
  Armando Broja              0.0
--------------------------------------------------
```

Hình 2.1: Chỉ số Standard SoT/90

```
Statistic: Total Cmp
====================
Top 3 players:
         Player  Total Cmp
Virgil van Dijk       2371
 William Saliba       2273
 Joško Gvardiol       2240

Bottom 3 players:
         Player  Total Cmp
  William Osula         11
Albert Grønbaek         13
Odsonne Édouard         17
--------------------------------------------------
```

Hình 2.2: Chỉ số Total Cmp

```
Statistic: Touches
==================
Top 3 players:
         Player  Touches
Virgil van Dijk     2915
 Joško Gvardiol     2808
   Levi Colwill     2669

Bottom 3 players:
         Player  Touches
Albert Grønbaek       38
  William Osula       41
Odsonne Édouard       45
------------------------------------------------
```

Hình 2.3: Chỉ số Touches

```
Statistic: Tackles Tkl
======================
Top 3 players:
            Player  Tackles Tkl
Idrissa Gana Gueye           123
      Daniel Muñoz           109
    Moisés Caicedo           100

Bottom 3 players:
          Player  Tackles Tkl
  Aaron Ramsdale             0
Łukasz Fabiański             0
 Bart Verbruggen             0
--------------------------------------------------
```

Hình 2.4: Chỉ số Tackles Tkl

```
Statistic: Tackles TklW
=======================
Top 3 players:
            Player  Tackles TklW
Idrissa Gana Gueye            74
      Daniel Muñoz            70
    Moisés Caicedo            63

Bottom 3 players:
          Player  Tackles TklW
  Aaron Ramsdale             0
Łukasz Fabiański             0
 Bart Verbruggen             0
--------------------------------------------------
```

Hình 2.5: Chi số Tackles Tklw

```
Statistic: Performance Recov
============================
Top 3 players:
         Player  Performance Recov
 Moisés Caicedo                197
Bruno Fernandes                187
  Ryan Christie                182

Bottom 3 players:
                 Player  Performance Recov
         Fraser Forster                  1
             Danny Ward                  1
Hákon Rafn Valdimarsson                  1
--------------------------------------------------
```

Hình 2.6: Chi số Performance Recov

* Tính trung vị , độ lệch chuẩn

1. Cách triển khai

- Bước 1: Tính toán theo đội

```
results = []
for team, group_df in df.groupby('Squad'):
    row = [team]
    for col in numeric_columns:
        row.extend([
            group_df[col].median(),
            group_df[col].mean(),
            group_df[col].std()
        ])
    results.append(row)
```

- Nhóm dữ liệu theo Squad (đội bóng).

- Với mỗi đội: Tính median, mean, std (độ lệch chuẩn) cho từng chỉ số.

- Lưu vào results.

- Bước 2: Thêm dòng trung bình toàn giải

```
league_row = ['All players']
for col in numeric_columns:
    league_row.extend([
        df[col].median(),
        df[col].mean(),
        df[col].std()
    ])
results.append(league_row)
```

- Thêm dòng tổng hợp chung toàn giải (All players).

- Bước 3: Tạo các header

```
header = ['Squad']
for col in numeric_columns:
    header.extend([
        f"Median {col}",
        f"Mean {col}",
        f"StdDev {col}"
    ])
```

- Bước 4: Xuất file CVS thống kê

```
final_df = pd.DataFrame(results, columns=header)
final_df.to_csv(RESULTS_FILE, index=False, encoding=ENCODING)
print(f"Lưu thống kê vào '{RESULTS_FILE}'")
```

2. Kết quả

Squad	Median St	Mean Stan	StdDev Sta	Median To	Mean Tota	StdDev To	Median To	Mean Tou	StdDev To	Median Ta	Mean Tacl	StdDev Ta	Median Ta	Mean Tacl	StdDev Ta	Median Pe	Mean Perf	StdDev Performance Recov
Arsenal	0.225	0.410909	0.410051	459.5	700.4091	551.7826	738.5	983.7727	662.5282	19	24.09091	20.32805	12	14.27273	12.32953	48	58.04545	40.02794
Aston Villa	0.19	0.449259	0.676774	470	489.963	414.9424	674	718.6296	558.2959	11	21.37037	20.34916	6	12.2963	12.42768	39	46.37037	35.55294
Bournemo	0.16	0.554348	0.659654	415	478.7391	387.9912	778	809.2174	595.0946	21	27.08696	22.79735	10	15.73913	14.20377	55	69.91304	55.65389
Brentford	0.17	0.31381	0.373544	471	553.8571	426.4507	762	889.0476	639.207	24	25.95238	21.28726	16	15.57143	12.48428	50	67.33333	53.9438
Brighton	0.27	0.333462	0.312921	390.5	539.5	477.7837	638	804.5	607.9543	17	23.65385	19.09333	11	14.38462	11.64844	47	55.15385	43.76454
Chelsea	0.315	0.474583	0.506694	512	656.2083	594.4997	798	915.25	770.5291	12.5	21.08333	23.89273	9	13.20833	14.79418	42.5	56.83333	50.54027
Crystal Pal	0.34	0.496667	0.466758	511	504.381	382.2344	827	860.8571	604.1604	21	32.61905	29.18985	14	19.14286	18.1226	60	70.85714	52.53978
Everton	0.21	0.335714	0.357471	434	467.7619	331.1142	734	804.8571	500.3479	27	30.95238	28.46836	14	18.19048	17.17737	60	63.7619	47.01054
Fulham	0.265	0.5345	0.620979	643	644	476.5614	950	954.85	646.3387	24	28.3	22.49	12	17.15	14.64052	58	65.85	44.65337
Ipswich To	0.07	0.267931	0.361271	208	344.6207	304.6264	433	575.4138	459.9646	12	18.03448	16.33245	8	10.17241	9.184883	37	43.72414	35.75801
Leicester C	0.21	0.2628	0.267138	430	489.84	367.6899	804	757.56	522.274	21	25.44	22.3869	11	14.92	13.43168	38	50.36	37.36741
Liverpool	0.35	0.510952	0.461041	590	756.0952	609.8724	758	1070.524	788.2621	24	27.14286	22.77781	15	16.66667	14.11855	58	66.71429	47.20185
Mancheste	0.33	0.4688	0.52528	676	767.72	566.6331	878	1000.36	670.0436	15	17.68	16.07462	7	10.8	10.43232	46	51.72	34.71253
Mancheste	0.21	0.387308	0.463418	561	551.8077	460.2912	767.5	812.4615	650.8429	20	27.46154	27.57714	9.5	16.46154	16.75048	43.5	57.07692	50.86211
Newcastle	0.28	0.371304	0.416973	459	599.4348	511.8292	747	898.1304	713.881	22	23.82609	21.35554	14	14.56522	13.34448	46	65	57.4393
Nott'ham	0.32	0.446667	0.439082	426	425.5238	320.3084	900	760.1905	527.4502	21	28.90476	24.91767	13	17.61905	15.4967	50	63.90476	50.94399
Southampt	0.15	0.217931	0.28481	242	485.4828	464.8155	457	707	616.1692	13	20	18.89066	7	11.51724	10.8121	23	45.03448	42.99376
Tottenhan	0.3	0.424815	0.432145	681	560.9259	367.0004	862	811.2963	510.7239	13	21.88889	20.23959	9	13.96296	13.71546	56	54.7037	40.28725
West Ham	0.23	0.3904	0.412538	510	519	397.9965	656	806.28	580.7711	17	22.52	19.92929	11	13.28	11.75982	43	55.52	46.6879
Wolves	0.23	0.36	0.413477	484	548.6957	399.8347	724	852.087	555.8876	19	30.86957	27.00893	12	18.13043	16.79909	43	64.17391	49.33987
All players	0.23	0.395303	0.454528	452	550.2109	451.2267	738	831.5491	610.0347	17	24.59499	22.27197	10	14.68476	13.67918	45	57.79123	45.76091

Hình 2.7: Bảng tính toán các chỉ số của các đội và toàn bộ cầu thủ

* Vẽ biểu đồ

1. Cách triển khai

- Vẽ biểu đồ theo toàn giải

```
data = df[col].dropna()
plt.figure(figsize=(10, 6))
if data.empty:
    plt.text(0.5, 0.5, 'No data available', ha='center', va='center')
else:
    plt.hist(data, bins=20, color='steelblue', alpha=0.7)
plt.title(f"Histogram of {col} - League")
plt.xlabel(col)
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig(os.path.join(stat_dir, "league.png"))
plt.close()
```

- Vẽ biểu đồ theo các đội

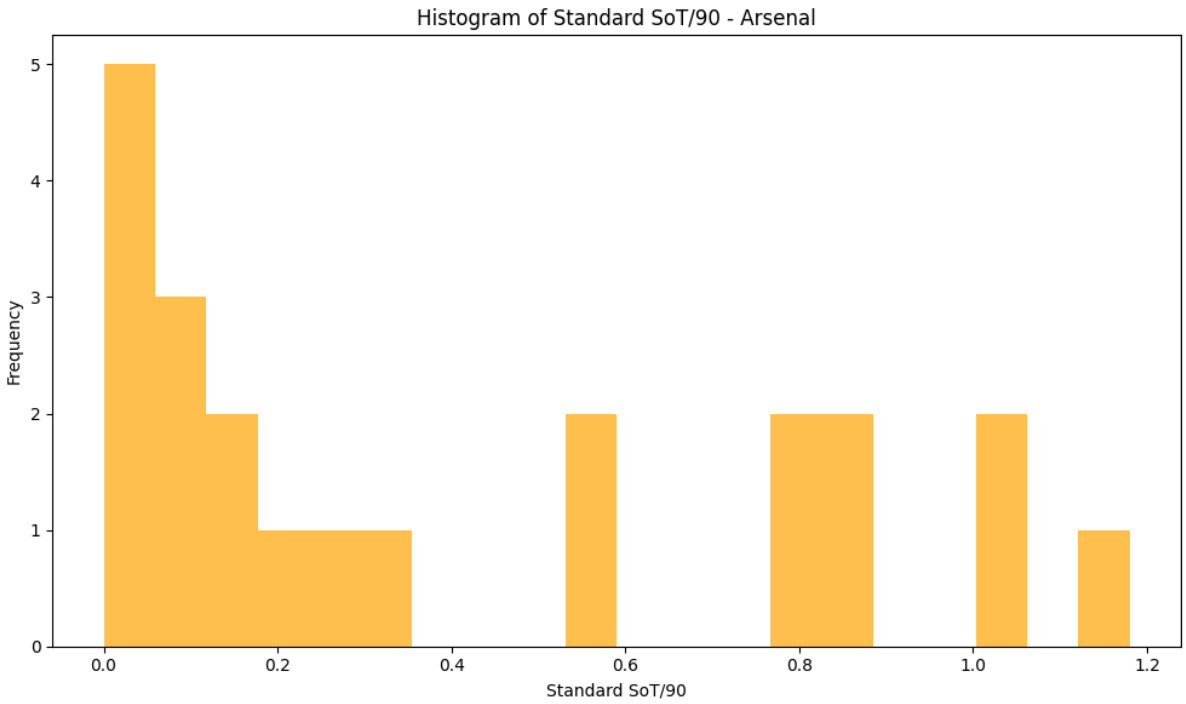
```
for team in df['Squad'].dropna().unique():
    team_data = df[df['Squad'] == team][col].dropna()
    plt.figure(figsize=(10, 6))
    if team_data.empty:
        plt.text(0.5, 0.5, 'No data available', ha='center', va='center')
    else:
        plt.hist(team_data, bins=20, color='orange', alpha=0.7)
    plt.title(f"Histogram of {col} - {team}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.tight_layout()
    safe_team = re.sub(r'[^\w]', '_', str(team))
    plt.savefig(os.path.join(stat_dir, f"{safe_team}.png"))
    plt.close()
```

- Với mỗi chỉ số:

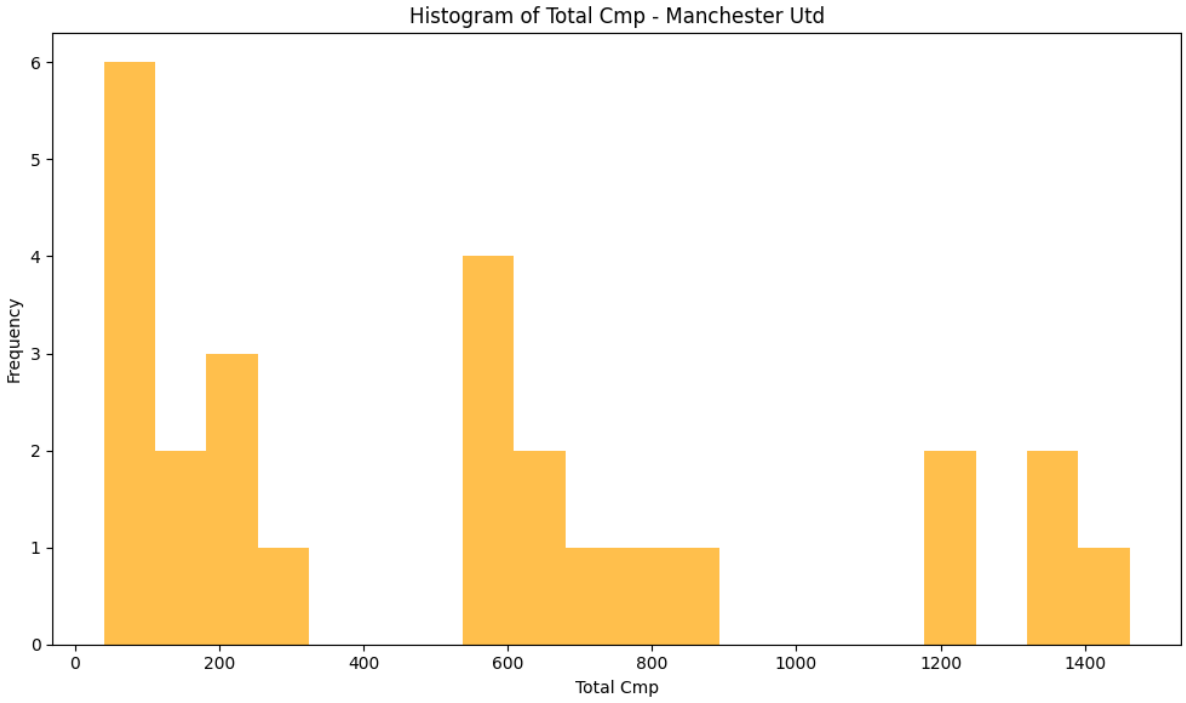
+ Tạo thư mục con theo tên chỉ số.
+ Biểu đồ toàn giải: vẽ histogram từ tất cả cầu thủ.
+ Biểu đồ theo đội: vẽ histogram riêng cho từng đội.
+ Nếu không có dữ liệu: ghi chú "No data available".
+ Lưu từng biểu đồ dưới dạng .png.

2. Kết quả

- Do có khá nhiều đội và kết quả nên ta chỉ lấy một vài ảnh



Hình 2.8: Chỉ số Standard SoT/90 đối với đội Arsenal



Hình 2.9: Chỉ số Total Cmp đối với đội Manchester United

Bình luận: Dựa trên phân tích số liệu, Chelsea hiện đang là đội bóng có phong độ ấn tượng nhất tại mùa giải Ngoại hạng Anh 2024-2025. Đội bóng thành London dẫn đầu ở các chỉ số tấn công then chốt như bàn thắng kỳ vọng (xG), kiến tạo kỳ vọng (xAG), các hành động tạo cơ hội ghi bàn (SCA), đường chuyền then chốt (KP) và đường chuyền vào khu vực nguy hiểm (PPA). Những con số này phản ánh khả năng tạo ra cơ hội ghi bàn chất lượng cao một cách thường xuyên. Không những vậy, Chelsea còn cho thấy sự vượt trội trong việc kiểm soát thế trận thông qua lượng chạm bóng và số đường chuyền vào các khu vực trọng yếu – cho thấy sự hiệu quả trong giai đoạn tấn công. Về khía

cạnh phòng ngự, mặc dù không dẫn đầu tuyệt đối, nhưng các chỉ số như số lần tắc bóng và cắt bóng của họ vẫn nằm trong nhóm các đội có thành tích tốt nhất.

Tổng kết lại, Chelsea đang thể hiện sự vượt trội trên nhiều phương diện và xứng đáng được đánh giá là đội bóng có màn trình diễn toàn diện nhất tại Premier League mùa giải năm nay.

III Bài Tập 3

1. Cách triển khai

Bước 1: Đọc dữ liệu từ file

```
try:
    df = pd.read_csv(CSV_FILE, encoding=ENCODING)
    print(f"Reading file '{CSV_FILE}' successfully.")
except Exception as e:
    print(f"Error reading file: {e}")
    exit()
```

- Đọc dữ liệu từ file results.csv (được xuất từ phân tích trước).
- Nếu có lỗi, chương trình sẽ thoát (exit()).

Bước 2: Lấy các cột số

```
numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()
if not numeric_columns:
    print("No columns to analyze.")
    exit()
```

- Lấy danh sách tất cả các cột có kiểu dữ liệu số (int, float).
- Nếu không có cột số nào thì dừng lại.

Bước 3: Chuẩn hóa dữ liệu

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[numeric_columns])
```

- Dữ liệu được chuẩn hóa (z - score: trung bình 0, độ lệch chuẩn 1) để các biến không bị ảnh hưởng bởi đơn vị đo khác nhau.
- Đây là bước bắt buộc trước khi dùng KMeans hoặc PCA.

Bước 4: Elbow Method để chọn số cụm

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

elbow_k = 3
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.axvline(x=elbow_k, color='red', linestyle='--', label=f'Elbow at k={elbow_k}')
plt.annotate('← The most obvious Elbow', xy=(elbow_k, inertia[elbow_k-1]),
             xytext=(elbow_k + 3, inertia[elbow_k-1] + 100),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
plt.title("Elbow chart - Select the optimal number of clusters")
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

- Inertia: tổng khoảng cách từ mỗi điểm đến tâm cụm — càng thấp, cụm càng "chặt".
- Elbow method sẽ tìm “k” tại đó inertia bắt đầu giảm chậm lại

Bước 5: Phân cụm với Kmeans

```
kmeans = KMeans(n_clusters=OPTIMAL_K, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)
```

- Dữ liệu được gán thêm một cột Cluster, biểu thị cụm (cluster) mà mỗi cầu thủ thuộc về.
- KMeans phân cụm các cầu thủ dựa trên các chỉ số đã chuẩn hóa.

Bước 6: Tóm tắt thống kê theo từng cụm

```
print("\nAverage of cluster indices:")
cluster_summary = df.groupby('Cluster')[numeric_columns].mean().round(2)
print(cluster_summary)
```

- Tính giá trị trung bình của từng chỉ số số học cho từng cụm.
- Giúp hiểu đặc điểm chung của mỗi nhóm cầu thủ.

Bước 7: Giảm chiều dữ liệu bằng PCA

```
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_data)
```

- PCA giúp giảm số chiều dữ liệu từ nhiều chỉ số còn 2 chiều để dễ vẽ.
- Các chiều mới là tổ hợp tuyến tính của các chỉ số gốc.

Bước 8: Trực quan hóa phân cụm bằng biểu đồ 2D

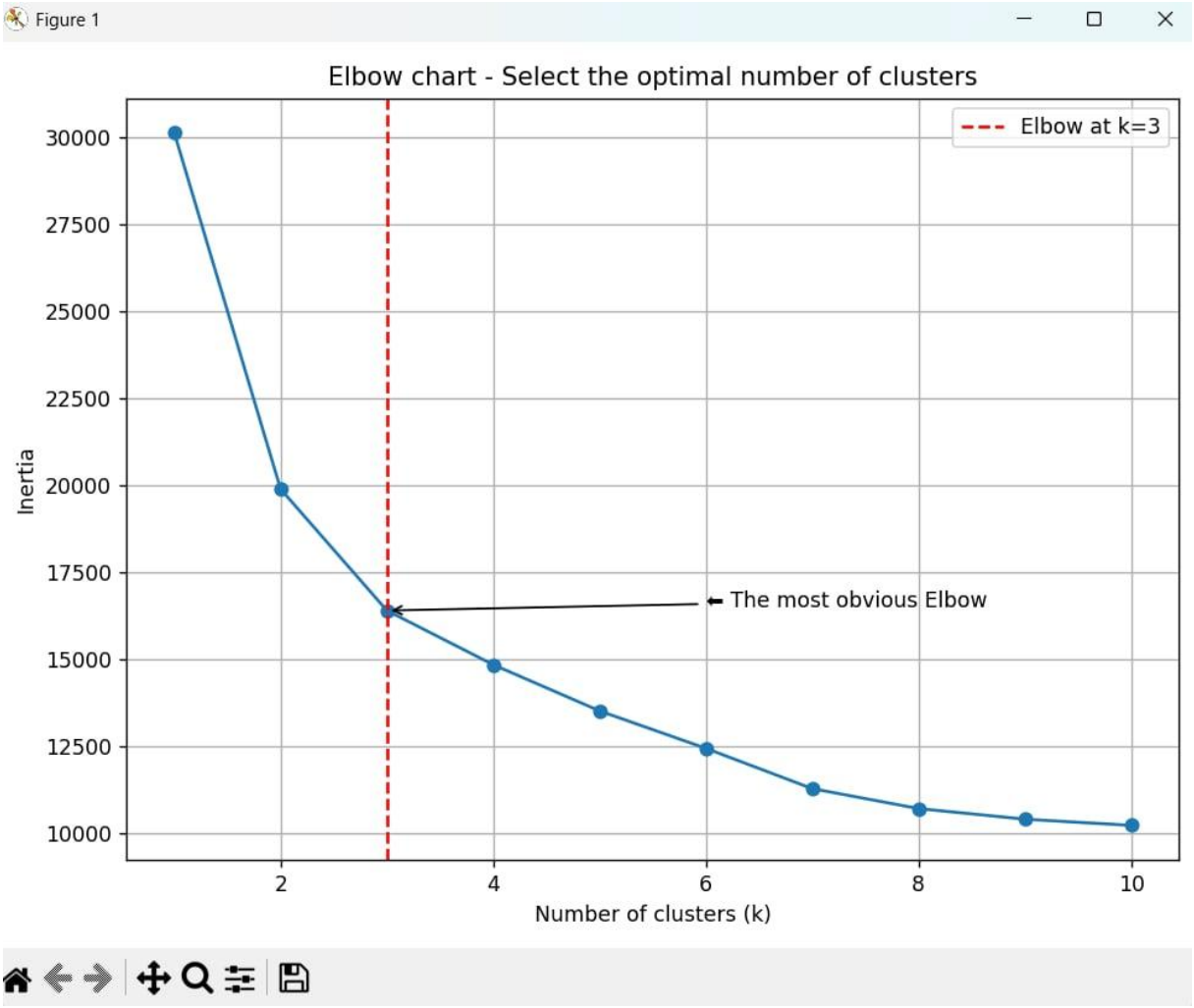
```
df_pca = pd.DataFrame(pca_components, columns=['PCA1', 'PCA2'])
df_pca['Cluster'] = df['Cluster']

plt.figure(figsize=(10, 8))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=df_pca,
                palette="viridis", s=100, alpha=0.7)
plt.title("Player clustering (PCA 2D)")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend(title="CLuster")
plt.grid(True)
plt.tight_layout()
plt.show()
```

- Vẽ các cầu thủ trên mặt phẳng 2D theo 2 thành phần PCA.
- Mỗi điểm là 1 cầu thủ, màu sắc thể hiện cụm mà họ thuộc vào.

2. Kết quả

Bình luận: Việc chọn số cụm k = 3 trong phương pháp KMeans là hợp lý vì, dựa trên thuật toán Elbow, ta quan sát thấy một điểm "khuỷu" rõ ràng tại k = 3. Khi vẽ đồ thị 'inertia' (tổng khoảng cách từ các điểm dữ liệu đến tâm cụm), ta nhận thấy rằng sau k = 3, mức giảm của inertia không còn đáng kể. Điều này cho thấy ba cụm đã phân chia tối ưu của dữ liệu, vì việc tăng số cụm vượt quá k = 3 sẽ không cải thiện đáng kể độ phân tán của các điểm.



Hình 3.1: Biểu đồ về số cụm tối ưu

Như ta có thể thấy trên biểu đồ Elbow, k = 2 cũng có thể là một cách phân cụm hợp lý, nhưng có một số lý do để không chọn phương án này, chẳng hạn như có thể bỏ qua một sự phân định trong dữ liệu, dẫn đến việc thiếu sự phân biệt rõ ràng giữa các nhóm cầu thủ nếu chọn k = 2. (chỉ có 2 nhóm)

- **Nhóm 0**: Cầu thủ thiên về phòng ngự (hậu vệ, tiền vệ phòng ngự, thủ môn, v.v...) – tham gia nhiều thời gian thi đấu nhưng ít tấn công.
- **Nhóm 1**: Cầu thủ dự bị, ít đóng góp.
- **Nhóm 3**: Cầu thủ tấn công với số bàn thắng/kiến tạo cao.

Hình 3.2: Biểu đồ về số cụm tương ứng

IV Bài Tập 4

1. Cách triển khai

Bước 1: Đọc và Kiểm Tra Dữ Liệu Input

```
try:
    df = pd.read_csv(INPUT_CSV, encoding='utf-8-sig')
except FileNotFoundError:
    print(f"Error: '{INPUT_CSV}' not found.")
    exit()
except Exception as e:
    print(f"Error loading CSV: {e}")
    exit()
```

- Đoạn mã này cố gắng đọc dữ liệu từ một file CSV có tên results.csv. Nếu file không tồn tại hoặc có lỗi khi đọc, chương trình sẽ in ra thông báo lỗi và dừng lại.

Bước 2: Kiểm Tra Các Cột Cần Thiết

```
required_cols = ['Player', 'Playing Time Min']
team_col = next((col for col in ['Team', 'Squad', 'team', 'TEAM'] if col in df.columns), None)
if not all(col in df.columns for col in required_cols) or not team_col:
    print(f"Error: Required columns missing. Found: {df.columns.tolist()}")
    exit()
```

- Đảm bảo rằng dữ liệu đầu vào có những cột cần thiết (Player và Playing Time Min). Nếu thiếu các cột này, chương trình sẽ dừng lại.

Bước 3: Tiền Xử Lý Dữ Liệu

```
df['Playing Time Min'] = pd.to_numeric(df['Playing Time Min'].astype(str).str.replace(',', ''), errors='coerce')
df = df.dropna(subset=['Playing Time Min'])
```

- Dữ liệu trong cột Playing Time Min được chuyển thành kiểu số sau khi loại bỏ dấu phẩy. Những dòng có giá trị không hợp lệ trong cột này sẽ bị loại bỏ.

Bước 4: Lọc Cầu Thủ Đủ Thời Gian Chơi (Trên 900 phút)

```
players_900 = df[df['Playing Time Min'] > 900][['Player', team_col, 'Playing Time Min']].rename(columns={team_col: 'Team'})
```

- Lọc các cầu thủ có thời gian thi đấu trên 900 phút và lưu lại các cột cần thiết, bao gồm tên cầu thủ và đội bóng.

Bước 5: Hàm scrape_transfer_values**: Thu Thập Giá Trị Chuyển Nhượng**

```python
def scrape_transfer_values(players_df, headless=True):
    options = Options()
    if headless:
        options.add_argument("--headless")
    options.add_argument("--ignore-certificate-errors")
    options.add_argument("--disable-blink-features=AutomationControlled")
    options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/91.0.4472.124")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")
    options.add_argument("--allow-insecure-localhost")
    options.add_experimental_option("excludeSwitches", ["enable-automation"])
    options.accept_insecure_certs = True
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
    driver.set_page_load_timeout(60)
    values = []

    for _, row in players_df.iterrows():
        player = row['Player']
        transfer_value = None
```

- Hàm này sử dụng Selenium để mở trình duyệt và truy cập các trang web chuyển nhượng (Transfermarkt và FootballTransfers) để thu thập giá trị chuyển nhượng của cầu thủ.

```python
for attempt in range(3):
    try:
        print(f"Attempting for {player}")
        driver.get(f"{TRANSFERMARKT_URL}{player.replace(' ', '+')}")
        link = WebDriverWait(driver, 20).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, "td.hauptlink a[href*='/profil/spieler/']"))
        )
        player_url = link.get_attribute('href')
        driver.get(player_url)
        transfer_value = WebDriverWait(driver, 20).until(
            EC.presence_of_element_located((By.CSS_SELECTOR, "a[href*='/marktwertverlauf/spieler/']"))
        ).text
        print(f"Scraped {player}: {transfer_value}")
        break
    except Exception as e:
        time.sleep(5)
```

- Đầu tiên thử lấy dữ liệu từ Transfermarkt

```
if not transfer_value:
    for attempt in range(3):
        try:
            print(f"Attempting {player}")
            driver.get(f"{FOOTBALLTRANSFERS_URL}{player.replace(' ', '+')}")
            transfer_value = WebDriverWait(driver, 20).until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "div.market-value"))  # Cần xác minh selector này
            ).text
            print(f"Scraped {player}: {transfer_value}")
            break
        except Exception as e:
            time.sleep(5)
    else:
        transfer_value = 'N/a'

values.append({'player_name': player, 'transfer_value': transfer_value})
time.sleep(3)
```

- Nếu thất bại, sẽ thử FootballTransfers.
- Dữ liệu thu thập được sẽ được lưu vào một DataFrame.

Lí do: Do FootballTransfers thường hạn chế bot hoặc gây khó khăn trong việc tự động thu thập dữ liệu, nên ta sẽ ưu tiên sử dụng trang Transfermarkt để lấy thông tin cầu thủ. Trong trường hợp không thể truy cập dữ liệu từ FootballTransfers, hệ thống sẽ tự động chuyển sang lấy dữ liệu từ Transfermarkt như một phương án thay thế.

Bước 6: Kết Hợp Dữ Liệu Giá Trị Chuyển Nhượng với Dữ Liệu Gốc

```
try:
    transfer_data = scrape_transfer_values(players_900, headless=True)
    transfer_data.to_csv(TRANSFER_CSV, index=False, encoding='utf-8-sig')
except Exception as e:
    print(f"Scraping failed: {e}. Using fallback data.")
    transfer_data = pd.DataFrame({'player_name': players_900['Player'], 'transfer_value': [np.nan] * len(players_900)})
    transfer_data.to_csv(TRANSFER_CSV, index=False, encoding='utf-8-sig')

merged_data = pd.merge(players_900, transfer_data, left_on='Player', right_on='player_name', how='left')
```

- Sau khi thu thập giá trị chuyển nhượng, dữ liệu này sẽ được ghép nối vào DataFrame players_900 dựa trên tên cầu thủ.

Bước 7: Xử Lý Dữ Liệu Giá Trị Chuyển Nhượng

```
def clean_transfer_value(value):
    if value == 'N/a' or pd.isna(value):
        return np.nan
    try:
        value = value.re (function) lower: Any £', '').strip()
        if 'm' in value.lower():
            return float(value.lower().replace('m', '')) * 1e6
        elif 'k' in value.lower():
            return float(value.lower().replace('k', '')) * 1e3
        return float(value)
    except:
        return np.nan
```

- Dữ liệu giá trị chuyển nhượng sẽ được làm sạch: nếu giá trị là "N/a" hoặc không hợp lệ, sẽ gán là N/a. Các ký tự như "€", "£", "m", "k" sẽ được chuyển đổi thành số tương ứng.

Bước 8: Chọn Các Đặc Trưng Để Dự Đoán

```
features = ['Age', 'Min', 'Gls', 'Ast', 'xG', 'xAG', 'PrgC', 'PrgP', 'PrgR', 'SoT%', 'SoT/90', 'Tkl', 'TklW
```

```
def get_numeric_columns(df, cols):
    numeric_cols = []
    for col in cols:
        if col in df.columns:
            try:
                df[col] = pd.to_numeric(df[col].astype(str).replace(',', ''), errors='coerce')
                if df[col].notna().sum() > 0:
                    numeric_cols.append(col)
            except:
                pass
    return numeric_cols
```

- Chọn các đặc trưng (features) từ dữ liệu để sử dụng trong mô hình dự đoán giá trị chuyển nhượng.

Bước 9: Xử Lý Các Cột Số Liệu

```
available_features = get_numeric_columns(df, features)
if not available_features:
    print("No valid features. Exiting.")
    exit()

df['transfer_value'] = merged_data['transfer_value']
```

```
valid_df = df.dropna(subset=available_features + ['transfer_value'])

X = valid_df[available_features]
y = valid_df['transfer_value']
players = valid_df['Player']
```

- Chuyển đổi các cột chứa giá trị chuỗi thành kiểu số và đảm bảo rằng tất cả các giá trị trong cột là hợp lệ.

Bước 10: Huấn Luyện Mô Hình Dự Đoán Giá Trị Chuyển Nhượng

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

- Dữ liệu được chuẩn hóa (scaling) và chia thành bộ huấn luyện và bộ kiểm tra (80-20%).

- Sử dụng mô hình RandomForestRegressor để huấn luyện mô hình dự đoán giá trị chuyển nhượng.

Bước 11: Đánh Giá Mô Hình

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MSE: {mse:.2f}, R^2: {r2:.2f}")
```

- Tính toán lỗi trung bình bình phương (MSE) và hệ số xác định (R^2) để đánh giá hiệu suất của mô hình.

Bước 12: Dự Đoán Giá Trị Chuyển Nhượng Của Tất Cả Cầu Thủ

```
full_pred = model.predict(X_scaled)
```

- Sử dụng mô hình để dự đoán giá trị chuyển nhượng cho tất cả các cầu thủ trong dataset.

Bước 13: Lưu Kết Quả Dự Đoán , Giá Trị Chuyển Nhượng Và In

```python
predictions = pd.DataFrame({
    'Player': players.values,
    'Actual_Value': y.values,
    'Predicted_Value': full_pred
})
predictions.to_csv(PREDICT_CSV, index=False, encoding='utf-8-sig')

print(f"Done. Check '{TRANSFER_CSV}', '{PREDICT_CSV}'.")
```

Bình luận: Để có thể lấy được giá trị chuyển nhượng của cầu thủ, ta sẽ dùng mô hình Random Forest Regressor. Để chọn đặc trưng, ta kết hợp giữa hiểu biết chuyên môn (như tuổi, vị trí, phong độ, hợp đồng…) và các phương pháp phân tích dữ liệu. Ta dùng phân tích tương quan để loại bỏ những biến dư thừa, sau đó dùng mô hình cây như Random Forest để đánh giá tầm quan trọng của từng đặc trưng.

Về mô hình, do bài toán có nhiều mối quan hệ phi tuyến tính và phức tạp, nên em chọn Random Forest Regressor vì nó xử lý tốt dữ liệu kiểu này và không cần tiền xử lý rườm rà.

2. Kết quả

player_na	transfer_value
Aaron Ran	€16.00m
Aaron Wa	€22.00m
Abdoulaye	€8.00m
Adam Arn	€14.00m
Adam Smi	€1.00m
Adam Wh	€35.00m
Adama Tr	€10.00m
Alejandro	€45.00m
Alex Iwobi	€25.00m
Alexander	€100.00m
Alexis Ma	€90.00m
Alisson	€25.00m
Alphonse	€10.00m
Amad Dial	€40.00m
Amadou C	€55.00m
Andreas P	€20.00m
Andrew R	€20.00m
André	N/a
André On	€32.00m
Anthony E	€35.00m
Anthony G	€65.00m
Antoine S	€35.00m

Hình 4.1: Một phần của bảng giá trị chuyển nhượng cầu thủ

Player	Actual_Va	Predicted_Value
Aaron Cre	16000000	15279000
Aaron Ran	22000000	35910000
Abdukodi	1000000	7819000
Adam Smi	25000000	36200000
Adam Wel	1E+08	62479000
Adam Wh	90000000	65120000
Albert GrÃ	10000000	30720000
Alejandro	40000000	38160000
Alex McCa	20000000	23440000
Alex Palme	20000000	24290000
Alexis Mac	35000000	35010000
Ali Al Ham	65000000	58530000
Alisson	35000000	37110000
Alphonse	35000000	34100000
Amad Dial	35000000	36340000
Amadou C	6000000	35190000
Andreas P	500000	45940000
Andrew Rc	5000000	17830000

Hình 4.2: Một phần của bảng dự đoán giá trị chuyển nhượng

V Tổng Kết

Qua quá trình thực hiện bài tập lớn này, em đã vận dụng tổng hợp các kiến thức lập trình Python, kỹ năng xử lý dữ liệu, phân tích thống kê và trực quan hóa để xây dựng một hệ thống hoàn chỉnh thu thập, phân tích và dự đoán giá trị cầu thủ tại giải Ngoại hạng Anh mùa 2024–2025.

Cụ thể, em đã thành công trong việc:

- Tự động hóa việc thu thập dữ liệu từ các trang web uy tín như FBref và Transfermarkt.
- Tiến hành xử lý dữ liệu để trích xuất các chỉ số chuyên sâu liên quan đến hiệu suất thi đấu.
- Áp dụng thuật toán K-Means để phân nhóm cầu thủ theo đặc điểm chuyên môn, sử dụng PCA để trực quan hóa dữ liệu đa chiều.
- Dự đoán giá trị chuyển nhượng cầu thủ thông qua mô hình Random Forest Regressor, cho kết quả đánh giá mô hình đáng tin cậy.

Thông qua quá trình làm bài, em không chỉ củng cố thêm kiến thức lập trình và khoa học dữ liệu, mà còn hiểu rõ hơn tầm quan trọng của việc tiền xử lý dữ liệu, lựa chọn mô hình phù hợp và phân tích kết quả một cách khách quan.

Tuy nhiên, bài làm vẫn còn một số hạn chế như quy mô dữ liệu chưa quá lớn và việc xử lý một số trường hợp ngoại lệ chưa thực sự tối ưu. Em sẽ tiếp tục học hỏi, nghiên cứu sâu hơn về các kỹ thuật nâng cao nhằm cải thiện chất lượng phân tích trong các dự án tiếp theo.

Em xin chân thành cảm ơn thầy Kim Ngọc Bách đã tận tình hướng dẫn và hỗ trợ trong suốt quá trình thực hiện bài tập lớn này.