HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CƠ BẢN I

BỘ MÔN TIN HỌC CƠ SỞ

BÁO CÁO BÀI TẬP LỚN

Giảng viên	**: Kim Ngọc Bách**
Họ và tên sinh viên	**: Nguyễn Việt Pháp**
Mã sinh viên	**: D22CQCN07-B**
Lớp	**: B22DCCN607**
Nhóm	**: 11**

Hà Nội – 2024

Mục lục

I. Bài 1

1. import thư viện

1.1. thư viện pandas

Pandas là một thư viện Python để phân tích dữ liệu. Được Wes McKinney khởi xướng vào năm 2008 do nhu cầu về một công cụ phân tích định lượng mạnh mẽ và linh hoạt, pandas đã phát triển thành một trong những thư viện Python phổ biến nhất. Nó có một cộng đồng những người đóng góp cực kỳ tích cực .

Các thao tác sử dụng pandas trong bài :

- Tạo DataFrame từ dictionary:

```
df = pd.DataFrame(ans)
```

- Gộp dữ liệu từ các DataFrames:

```
result=pd.merge(result,df1,on=common_columns,how='left')
```

- Chuyển đổi kiểu dữ liệu của cột minutes:

```
result['minutes'] = result['minutes'].apply( lambda x : int(''.join(x.split(','))))
```

- Lọc các hàng có giá trị minutes lớn hơn 90:

```
result = result [result['minutes']>90]
```

- đổi tên cột :

```
result.rename(columns={'goals_pens': 'non-Penalty Goals', 'pens_made': 'Penalty Goals'},inplace=True)
```

- xóa cột :

```
result.drop(labels=['goals','goals_assists','birth_year',
                    'pens_att','npxg_xg_assist','minutes_90s',
                    'matches','gk_games', 'gk_games_starts',
                     'gk_minutes'],axis=1,inplace=True)
```

- Lưu DataFrame kết quả vào tệp CSV:

```
result.to_csv('result.csv',index=False)
```

1.2. Thư viện BeautifulSoup (Bs4)

Thư viện **BeautifulSoup** (được gọi là bs4 khi import) là một công cụ Python mạnh mẽ dùng để phân tích và trích xuất dữ liệu từ các trang web. Nó cho phép xử lý và tìm kiếm các phần tử HTML phức tạp một cách đơn giản và hiệu quả. Trong đoạn code của bạn, BeautifulSoup được sử dụng để tìm và phân tích dữ liệu từ HTML của một trang web.

Các thao tác sử dụng **BeautifulSoup** trong bài :

- Khởi tạo BeautifulSoup:

```
soup = bs(r.content, 'html.parser')
```

- Tìm kiếm:

tìm phần tử đầu tiên (**find**)

```
table = soup.find('div', {'id': id_div})
```

tìm tất cả phần tử (**find_all**)

```
comment = table.find_all(string=lambda text: isinstance(text, Comment))
```

1.3. thư viện Requests

Thư viện requests trong Python là công cụ hữu ích để thực hiện các yêu cầu HTTP, giúp lấy dữ liệu từ các trang web và API một cách dễ dàng. Thư viện này cung cấp các phương thức để gửi yêu cầu HTTP như GET, POST, PUT, DELETE. Trong đoạn mã của bạn, thư viện requests được sử dụng để tải nội dung HTML của một trang web, sau đó chuyển tiếp nội dung này đến BeautifulSoup để phân tích dữ liệu.

các thao tác sử dụng thư viện **requests**:

- Lấy nội dung của url :

```
r = requests.get(url)
```

- Truy xuất nội dung:

```
soup = bs(r.content, 'html.parser')
```

2. Triển khai

- viết hàm **crawler** để cào dữ liệu đối với mỗi **url** và **div:**

```
# hàm thu thập dữ liệu từ trang web
def crawler(url, id_div):
    print(url,id_div)
    r = requests.get(url)
    soup = bs(r.content, 'html.parser')
    table = soup.find('div', {'id': id_div})
    comment = table.find_all(string=lambda text: isinstance(text, Comment))
    data = bs(comment[0], 'html.parser').find_all('tr')
    # Tạo dictionary để lưu trữ dữ liệu được lấy ra
    ans = {}
    # lấy tên cột
    for i, g in enumerate(data[1].find_all('th')):
        if i !=0:
            ans[g.get('data-stat')] = []
    # lấy dữ liệu
    for i in range(2, len(data)):
        tmp = data[i].find_all('td')
        for j, x in enumerate(tmp):
            if x.get('data-stat') in ans.keys():
                if x.get('data-stat') == 'nationality':
                    s = x.getText().split(" ")
                    ans[x.get('data-stat')].append(s[0])
                else:
                    ans[x.get('data-stat')].append(x.getText())
    df = pd.DataFrame(ans)
    return df
```

- sau đó lặp hàm **crawler** vào gộp dữ liệu lại ở hàm **main:**

```
if __name__=='__main__':
    # Danh sách các URL cần lấy dữ liệu
    urls = [
            'https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/keepers/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/shooting/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/passing/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/passing_types/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/gca/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/defense/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/possession/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/playingtime/2023-2024-Premier-League-Stats',
            'https://fbref.com/en/comps/9/2023-2024/misc/2023-2024-Premier-League-Stats'
            ]
    # Các ID của thẻ div tương ứng cho mỗi URL
    ids = ['all_stats_standard','all_stats_keeper',
           'all_stats_shooting', 'all_stats_passing',
           'all_stats_passing_types', 'all_stats_gca',
           'all_stats_defense','all_stats_possession',
           'all_stats_playing_time', 'all_stats_misc']

    # Thu thập dữ liệu
    result=crawler(urls[0],ids[0])
    # Lần lượt gộp dữ liệu từ các URL khác
    df1=crawler(urls[1],ids[1])
    common_columns = list(result.columns.intersection(df1.columns))
    result=pd.merge(result,df1,on=common_columns,how='left')
    for i in range(2,len(urls)):
        tmp_df=pd.DataFrame(crawler(urls[
        tmp_common_columns = list(result.
        result=pd.merge(result,tmp_df,on=tmp_common_columns,how='inner')
    #  lọc các hàng có giá trị minutes lớn hơn 90.
    result['minutes'] = result['minutes'].apply( lambda x : int(''.join(x.split(','))))
    result = result [result['minutes']>90]
    # Đổi tên và loại bỏ các cột không cần thiết:
    result.rename(columns={'goals_pens': 'non-Penalty Goals', 'pens_made': 'Penalty Goals'},inplace=True)
    result.drop(labels=['goals','goals_assists','birth_year',
                        'pens_att','npxg_xg_assist','minutes_90s',
                        'matches','gk_games', 'gk_games_starts',
                         'gk_minutes'],axis=1,inplace=True)

    print(result.head().to_string())
    # Lưu kết quả vào CSV:
    result.to_csv('result.csv',index=False)
```

(variable) tmp_common_columns: list[str]

II. Bài 2

1.Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số

Thuật toán:

- với mỗi chỉ số, sắp xếp giảm dần
- lấy 3 số đầu là giá trị cao nhất
- lấy 3 số cuối là giá trị thấp nhất

Triển khai:

```
import matplotlib.pyplot as plt
import pandas as pd
import os

# Tạo thư mục để lưu trữ các bảng top 3
output_dir = 'top_3_tables'
os.makedirs(output_dir, exist_ok=True)

# Tạo dictionary để lưu top 3 giá trị cao nhất và thấp nhất cho mỗi chỉ số
top_3_highest = {}
top_3_lowest = {}

# Duyệt qua từng cột (chỉ số) trong DataFrame, bỏ qua cột đầu tiên ('player')
for col in df.columns[1:]:
    df_sorted = df.sort_values(by=col, ascending=False)
    top_3_highest[col] = df_sorted[['player', col]].head(3)
    top_3_lowest[col] = df_sorted[['player', col]].tail(3)

# Lưu top 3 cầu thủ có điểm cao nhất cho mỗi chỉ số
for col, top in top_3_highest.items():
    plt.figure(figsize=(6, 2))
    plt.axis('tight')
    plt.axis('off')
    plt.title(f'Top 3 cầu thủ có điểm cao nhất ở chỉ số: {col}')
    table_data = top.values
    columns = ['Cầu thủ', col]
    plt.table(cellText=table_data, colLabels=columns, cellLoc='center', loc='center')

    # Lưu hình
    filename = f"{output_dir}/top_3_highest_{col}.png"
    plt.savefig(filename, bbox_inches='tight')
    plt.close()  # Đóng hình để tiết kiệm bộ nhớ

# Lưu top 3 cầu thủ có điểm thấp nhất cho mỗi chỉ số
for col, bottom in top_3_lowest.items():
    plt.figure(figsize=(6, 2))
    plt.axis('tight')
    plt.axis('off')
    plt.title(f'Top 3 cầu thủ có điểm thấp nhất ở chỉ số: {col}')
    table_data = bottom.values
    columns = ['Cầu thủ', col]
    plt.table(cellText=table_data, colLabels=columns, cellLoc='center', loc='center')

    # Lưu hình
    filename = f"{output_dir}/top_3_lowest_{col}.png"
    plt.savefig(filename, bbox_inches='tight')
    plt.close()  # Đóng hình để tiết kiệm bộ nhớ
```

output : VD 2 bảng

Top 3 cầu thủ có điểm cao nhất ở chỉ số: aerials_won

Cầu thủ	aerials_won
Virgil van Dijk	140
James Tarkowski	140
Dominic Calvert-Lewin	137

Top 3 cầu thủ có điểm cao nhất ở chỉ số: aerials_lost

Cầu thủ	aerials_lost
Carlton Morris	169
Dominic Calvert-Lewin	144
Dominic Solanke	135

...

2.Tìm trung vị, trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội

Thuật toán:

- dùng **median()** để tìm trung vị:

```
table['Median of ' + att].append(float(numeric_df[att].median()))
```

- dùng **mean()** để tìm trung bình:

```
table['Mean of ' + att].append(float(numeric_df[att].mean()))
```

- dùng **std()** để tìm độ lệch chuẩn :

```
table['Std of ' + att].append(float(numeric_df[att].std()))
```

Triển khai:

```
# Hàm thêm giá trị thống kê cho từng team
def add_statistics_for_team(team_name, numeric_df, table):
    table['Team'].append(team_name)
    for att in numeric_df.columns:
        table['Median of ' + att].append(float(numeric_df[att].median()))
        table['Mean of ' + att].append(float(numeric_df[att].mean()))
        table['Std of ' + att].append(float(numeric_df[att].std()))

if __name__=='__main__':
    numeric_df = df.select_dtypes(include=['float', 'int'])
    # Tạo dictionary để lưu kết quả
    table = {'Team': []}
    # Khởi tạo các cột cho Median, Mean, Std của từng thuộc tính số
    for att in numeric_df.columns:
        table['Median of ' + att] = []
        table['Mean of ' + att] = []
        table['Std of ' + att] = []
    # Tính toán cho tất cả các đội (team 'all')
    add_statistics_for_team('all', numeric_df, table)

    teams=['all']
    teams.extend(df['team'].unique())

    # Tính toán cho từng đội
    for team in teams[1:]:
        # Bỏ qua 'all' vì đã tính toán trước
        filtered_df = df[df['team'] == team]
        numeric_df = filtered_df.select_dtypes(include=['float', 'int'])
        add_statistics_for_team(team, numeric_df, table)

    # Tạo DataFrame từ dictionary 'table'
    result2 = pd.DataFrame(table)

    # lưu vào result2.csv
    result2.to_csv('result2.csv', index=False)
```

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

3.1. import thư viện

Thư viện matplotlib, đặc biệt là matplotlib.pyplot, là một trong những thư viện phổ biến nhất trong Python để tạo các biểu đồ trực quan hóa dữ liệu. matplotlib.pyplot được sử dụng để vẽ biểu đồ histogram (biểu đồ tần suất) nhằm trực quan hóa dữ liệu.

- vẽ biểu đồ histogram với các thông số:

- plt.figure(figsize=(5, 3)): Tạo một biểu đồ mới với kích thước xác định là 5x3 inches.
- plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black'): Vẽ histogram cho cột dữ liệu hiện tại (att) trong dataframe numeric_df.
- bins=10: Chia dữ liệu thành 10 khoảng giá trị (bins).
- alpha=0.7: Đặt độ trong suốt cho các cột của histogram để có thể dễ dàng nhìn rõ các cột chồng lên nhau.
- color='blue': Màu xanh dương cho cột trong biểu đồ.
- edgecolor='black': Đặt màu viền đen cho các cột để dễ dàng phân biệt giữa các khoảng.
- plt.title(f'Histogram of {att} (All Teams)'): Đặt tiêu đề cho biểu đồ, trong đó att là tên của thuộc tính hiện tại.
- plt.xlabel(att) và plt.ylabel('Frequency'): Gắn nhãn cho trục x và trục y.
- plt.grid(axis='y', alpha=0.75): Hiển thị lưới cho trục y, giúp dễ dàng quan sát giá trị tần suất.
- plt.show(): Hiển thị biểu đồ histogram.

```
plt.figure(figsize=(5, 3))
plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black')
plt.title(f'Histogram of {att} (All Teams)')
plt.xlabel(att)
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

3.2.Triển khai

```
import matplotlib.pyplot as plt

# lấy thuộc tính có giá trị float với int từ df
numeric_df=df.select_dtypes(include=['float','int'])
# print(df.head().to_string())
cnt=0
# Vẽ histogram cho toàn bộ giải đấu
for att in numeric_df.columns:
    cnt+=1
    print("Histogram: ",cnt)
    plt.figure(figsize=(5, 3))
    plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black')
    plt.title(f'Histogram of {att} (All Teams)')
    plt.xlabel(att)
    plt.ylabel('Frequency')
    plt.grid(axis='y', alpha=0.75)
    plt.show()

# Vẽ histogram cho mỗi đội
teams = df['team'].unique()

for team in teams:
    #  lọc cầu thủ cho mỗi đội
    filtered_df = df[df['team'] == team]
    # lấy thuộc tính có giá trị float với int từ df của mỗi đội
    numeric_df_team = filtered_df.select_dtypes(include=['float', 'int'])
    # Vẽ histogram cho toàn bộ giải đấu
    for att in numeric_df_team.columns:
        cnt+=1
        print("Histogram: ",cnt)
        plt.figure(figsize=(5, 3))
        plt.hist(numeric_df_team[att], bins=10, alpha=0.7, color='green', edgecolor='black')
        plt.title(f'Histogram of {att} ({team})')
        plt.xlabel(att)
        plt.ylabel('Frequency')
        plt.grid(axis='y', alpha=0.75)
        plt.show()
```

✓ 8m 20.8s

4. Tìm đội bóng có điểm cao nhất ở mỗi chỉ số , đánh giá đội có phong độ tốt nhất 2023-2024

4.1.Tìm đội bóng có điểm cao nhất ở mỗi chỉ số

thuật toán:

- tạo một danh sách lưu kết quả
- tìm chỉ số cao nhất ở mỗi cột bằng **idmax():**

```
max_idx = df2[column].idxmax()
```

- sau đó ghép dữ liệu rồi in kết quả

Triển khai:

```
df2=pd.read_csv('result2.csv')
df2.drop(index=0,inplace=True)
numeric_columns = df2.select_dtypes(include=['float', 'int']).columns

# Tạo danh sách để lưu kết quả cho mỗi thuộc tính
highest_team_per_stat = []

for column in numeric_columns:
    # Tìm chỉ số dòng có giá trị lớn nhất cho thuộc tính
    max_idx = df2[column].idxmax()

    # Lấy tên đội bóng và giá trị của thuộc tính tại chỉ số này
    max_team = df2.loc[max_idx, 'Team']
    max_score = df2.loc[max_idx, column]

    # Thêm kết quả vào danh sách
    highest_team_per_stat.append({
        'Attribute': column,
        'Team': max_team,
        'Highest Score': max_score
    })

# Chuyển đổi danh sách kết quả thành DataFrame
highest_team_per_stat_df = pd.DataFrame(highest_team_per_stat)

# Hiển thị kết quả
print(highest_team_per_stat_df)
```

output :

	Attribute	Team	Highest Score
0	Median of age	West Ham	27.500000
1	Mean of age	West Ham	28.272727
2	Std of age	Brighton	5.698324
3	Median of games	Fulham	29.000000
4	Mean of games	Fulham	27.238095
5	Std of games	Wolves	11.930773
6	Median of games_starts	Manchester City	24.000000
7	Mean of games_starts	Fulham	19.904762
8	Std of games_starts	Everton	13.720099
9	Median of minutes	Manchester City	2042.000000
10	Mean of minutes	Manchester City	1785.571429
11	Std of minutes	Everton	1156.480384
12	Median of assists	Liverpool	2.000000
13	Mean of assists	Manchester City	3.238095
14	Std of assists	Manchester City	3.780275
15	Median of non-Penalty Goals	Fulham	2.000000
16	Mean of non-Penalty Goals	Manchester City	4.047619
17	Std of non-Penalty Goals	Manchester City	5.757397
18	Median of Penalty Goals	Bournemouth	0.000000
19	Mean of Penalty Goals	Arsenal	0.476190

4.2. đánh giá đội có phong độ tốt nhất 2023-2024

Thuật toán :

- dựa vào (4.1) chọn ra đội có số chỉ số cao nhất nhiều nhất

Triển khai:

```
# đếm  xem mỗi đội bóng có bao nhiêu chỉ số điểm cao nhất
print(highest_team_per_stat_df['Team'].value_counts())
```

output :

```
Team
Manchester City    130
Liverpool           58
Fulham              50
Arsenal             37
West Ham            33
Everton             30
Bournemouth         30
Tottenham           21
Newcastle Utd       17
Luton Town          14
Sheffield Utd       13
Crystal Palace      13
Wolves              11
Aston Villa         10
Manchester Utd       9
Chelsea              8
Brentford            7
Nott'ham Forest      4
Burnley              2
Brighton             1
```

=> sau khi chạy đoạn mã thì ta thấy **Manchester City** là đội bóng có phong độ tốt nhất với **130** chỉ số điểm cao nhất.

III. Bài 3

1. Sử thuật toán Kmeans để phân loại cầu thủ

1.1. Sử dụng thuật toán Elbow để chọn số cụm phần loại

import thư viện

- thư viện **sklearn.cluster.KMeans**

lặp cho từng giá trị của k (số cụm):

- KMeans(n_clusters=i, init='k-means++', random_state=42):
  - n_clusters=i: xác định số lượng cụm cần phân chia
  - init='k-means++' tối ưu hóa việc chọn tâm cụm ban đầu
  - random_state=42 đảm bảo tính tái lập của kết quả.

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)
```

- thư viện **sklearn.preprocessing**

sklearn.preprocessing.StandardScaler: (dạng float) Chuẩn hóa dữ liệu bằng cách đưa mỗi đặc trưng về phân phối chuẩn có trung bình bằng 0 và độ lệch chuẩn bằng 1

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(tmp_df_transformed)
```

sklearn.preprocessing.OneHotEncoder: (dùng cho các cột dạng String)

OneHotEncoder tạo ra các cột nhị phân tương ứng với từng giá trị duy nhất trong cột nationality, position, và team.

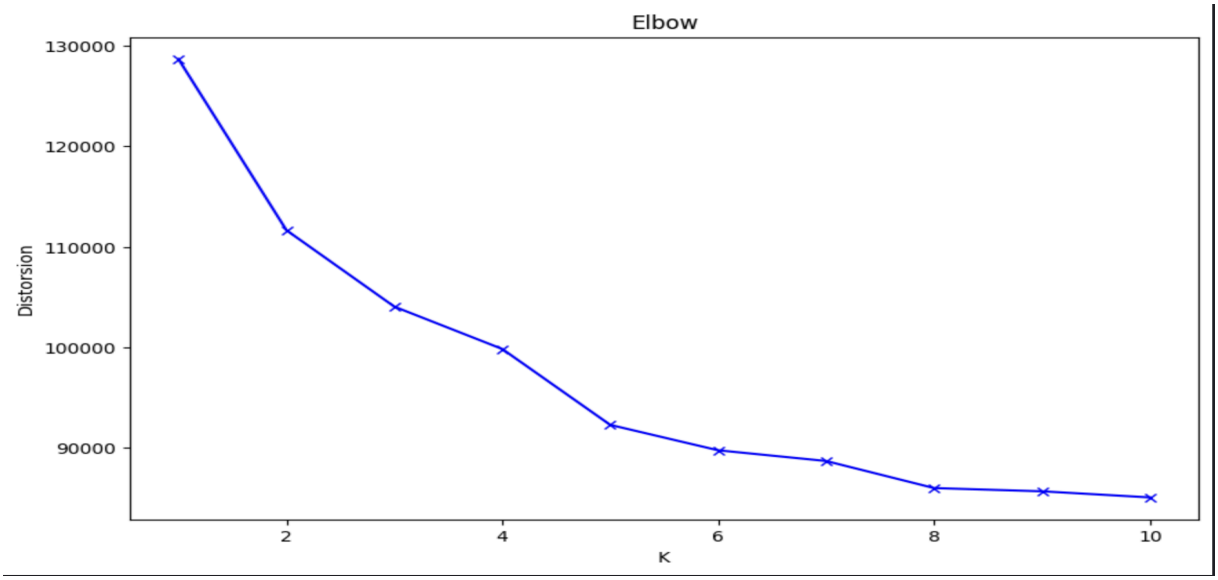
```
ct = ColumnTransformer(
    transformers=[('encode', OneHotEncoder(), ['nationality', 'position', 'team'])],
    remainder='passthrough'
)
transformed_data = ct.fit_transform(tmp_df)
```

Triển khai

Thuật toán **Elbow** là phương pháp dùng để xác định số lượng cụm tối ưu trong phân cụm K-Means. Mục tiêu của phương pháp là tìm số lượng cụm k mà tại đó, việc tăng số cụm sẽ không giảm đáng kể độ méo mó (distortion) của mô hình.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

# Xóa cột không cần thiết
tmp_df = df.drop(columns='player')
# Điền giá trị thiếu bằng 0
tmp_df = tmp_df.fillna(0)
# Chuyển đổi dữ liệu dạng String thành one-hot encoding
ct = ColumnTransformer(
    transformers=[('encode', OneHotEncoder(), ['nationality', 'position', 'team'])],
    remainder='passthrough'
)
transformed_data = ct.fit_transform(tmp_df)

encoded_columns = ct.named_transformers_['encode'].get_feature_names_out(['nationality', 'position', 'team'])
all_columns = list(encoded_columns) + [col for col in tmp_df.columns if col not in ['nationality', 'position', 'team']]
tmp_df_transformed = pd.DataFrame(transformed_data, columns=all_columns)

scaler = StandardScaler()
df_scaled = scaler.fit_transform(tmp_df_transformed)

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)


plt.figure(figsize=(10,6))
plt.plot(range(1, 11), wcss,'bx-')
plt.xlabel("K")
plt.ylabel("Distorsion")
plt.title("Elbow")
plt.show()
```

biểu đồ :



với biểu đồ trên, ta cần chọn **Điểm *khuỷ tay*** là điểm mà ở đó tốc độ suy giảm của *hàm biến dạng* sẽ thay đổi nhiều nhất, Tức là kể từ sau vị trí này thì gia tăng thêm số lượng cụm cũng không giúp *hàm biến dạng* giảm đáng kể=> ta **chọn số cụm(k)=5**

1.2. Phần cụm dữ liệu bằng thuật toán Kmeans

- xác định số cụm
- Khởi tạo mô hình KMeans
- Dự đoán cụm cho dữ liệu
- Thêm nhãn cụm vào DataFrame

Triển Khai

```
n_clusters=5
kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
y_kmeans = kmeans.fit_predict(df_scaled)
tmp_df_transformed['Cluster'] = y_kmeans
print(tmp_df_transformed.head().to_string())
```

2. dùng PCA giảm chiều dữ liệu, vẽ hình phân cụm

Principal Component Analysis (PCA) là một kỹ thuật giảm chiều dữ liệu thường được sử dụng trong học máy và thống kê. Mục tiêu của PCA là giảm số lượng chiều của dữ liệu trong khi giữ lại phần lớn thông tin cần thiết, giúp đơn giản hóa và tăng hiệu quả của các thuật toán học máy mà không làm mất quá nhiều thông tin.

2.1. import thư viện

- from **sklearn.decomposition** import **PCA**
- khởi tạo PCA

```
pca = PCA(n_components=2)
```

- áp dụng PCA

```
data_pca = pca.fit_transform(df_scaled)
```

2.2. triển khai

giảm chiều với PCA

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Áp dụng PCA
pca = PCA(n_components=2)
data_pca = pca.fit_transform(df_scaled)

# Tạo DataFrame với các thành phần chính và nhãn cụm
df_pca = pd.DataFrame(data_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = tmp_df_transformed['Cluster']
```

vẽ hình phân cụm:

```
# Vẽ biểu đồ các cụm
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b','c','m']
for i in range(n_clusters):
    plt.scatter(df_pca[df_pca['Cluster'] == i]['PC1'],
                df_pca[df_pca['Cluster'] == i]['PC2'],
                s=100,
                c=colors[i],
                label=f'Cluster {i}')

plt.title('K-Means Clustering with PCA')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid()
plt.show()
```

3. vẽ biểu đồ radar so sánh cầu thủ

các bước tạo biểu đồ radar:

- Chuẩn bị dữ liệu và góc cho các trục:

```
players=[p1,p2]
N = len(Attributes)

# Tính góc cho từng trục
theta = [n / float(N) * 2 * pi for n in range(N)]
theta += theta[:1]  # Đóng vòng tròn cho các góc
```

- Tạo giá trị cho từng cầu thủ và đóng vòng tròn

```
# Tạo giá trị cho từng cầu thủ và đóng vòng tròn
values = []
for player in players:
    player_values = radar_df.loc[radar_df['player'] == player, Attributes].values.flatten().tolist()
    values.append(player_values + player_values[:1])  # Đóng vòng tròn
```

- tạo biểu đồ

```
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
colors = ['b', 'r']
```

- Vẽ dữ liệu cho từng cầu thủ

```
for i, player_values in enumerate(values):
    ax.plot(theta, player_values, color=colors[i], linewidth=2, label=players[i])
    ax.fill(theta, player_values, facecolor=colors[i], alpha=0.25)
```

- Thiết lập các trục và tiêu đề

```
# Thiết lập các trục và tiêu đề
ax.set_xticks(theta[:-1])
ax.set_xticklabels(Attributes, fontsize=10)
ax.set_title('Player Attribute Comparison', weight='bold', size=16, position=(0.5, 1.1))
ax.set_rgrids([2*max_value/10, 4*max_value/10, 6*max_value/10, 8*max_value/10, 10*max_value/10])  # Điều chỉnh các giá trị trục theo nhu cầu

# Thêm chú thích
ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
```

Triển khai:

nhận yêu cầu từ input:

```
print("Danh sách tên cầu thủ: ",list(df['player']),sep='\n')
p1=input("Mời bạn nhập tên cầu thủ thứ nhất: ").strip()
p2=input("Mời bạn nhập tên cầu thủ thứ hai: ").strip()
list_att=list(df.drop(columns=['player','nationality','position','team'],axis=1).columns)
print("Danh sách các thuộc tính: ",list_att,sep='\n')
Attributes=list(i for i in input("Mời bạn nhập những thuộc tính cần so sánh: ").split())
```

✓ 48.4s

vẽ radar:

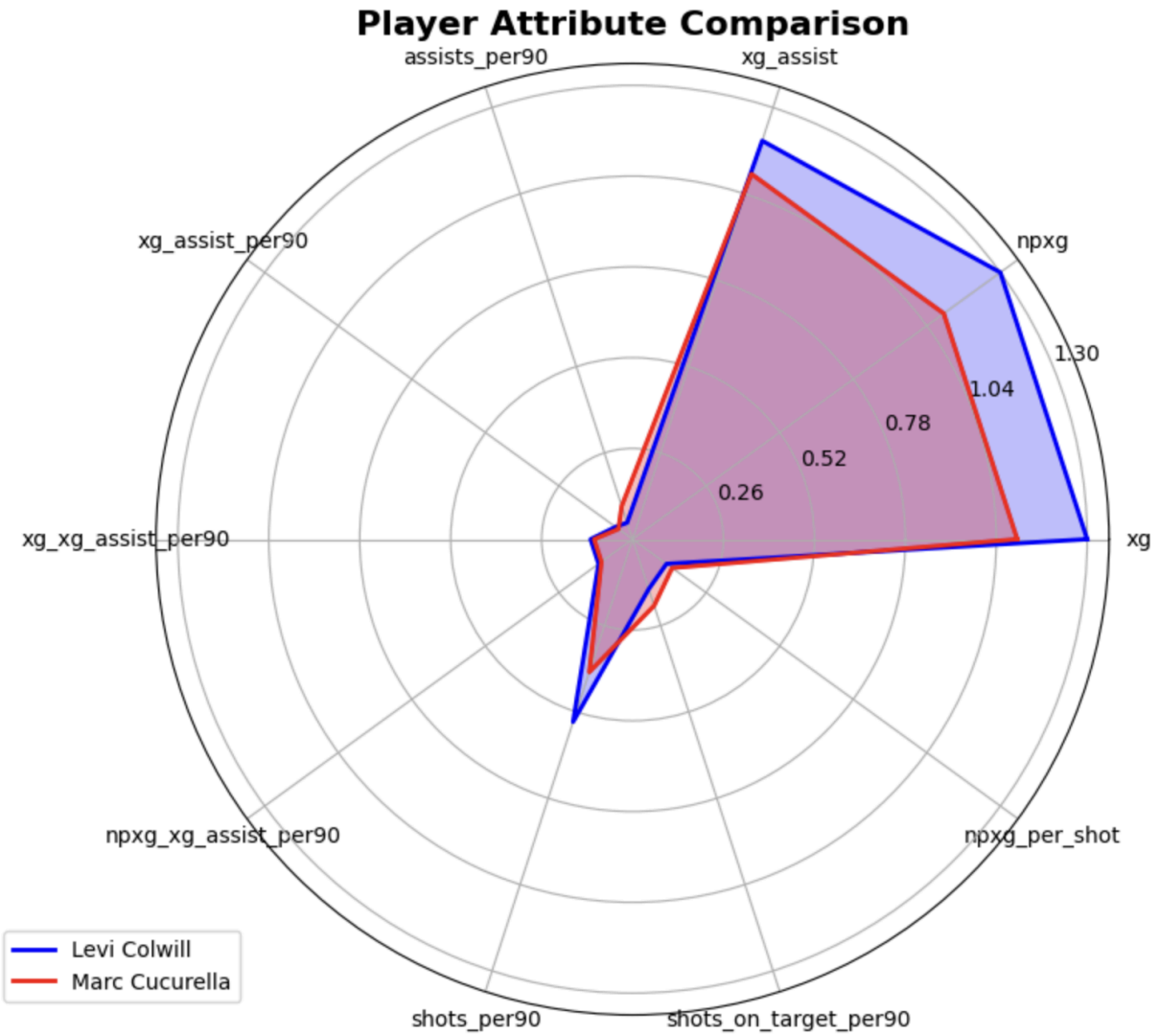
```python
import numpy as np
import matplotlib.pyplot as plt
from math import pi

players=[p1,p2]
N = len(Attributes)

# Tính góc cho từng trục
theta = [n / float(N) * 2 * pi for n in range(N)]
theta += theta[:1]  # Đóng vòng tròn cho các góc
# Tạo giá trị cho từng cầu thủ và đóng vòng tròn
values = []
for player in players:
    player_values = radar_df.loc[radar_df['player'] == player, Attributes].values.flatten().tolist()
    values.append(player_values + player_values[:1])  # Đóng vòng tròn

# Tạo biểu đồ radar
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
colors = ['b', 'r']

# Vẽ dữ liệu cho từng cầu thủ
for i, player_values in enumerate(values):
    ax.plot(theta, player_values, color=colors[i], linewidth=2, label=players[i])
    ax.fill(theta, player_values, facecolor=colors[i], alpha=0.25)

max_value = radar_df.drop(columns='player').max().max()

# Thiết lập các trục và tiêu đề
ax.set_xticks(theta[:-1])
ax.set_xticklabels(Attributes, fontsize=10)
ax.set_title('Player Attribute Comparison', weight='bold', size=16, position=(0.5, 1.1))
ax.set_rgrids([2*max_value/10, 4*max_value/10, 6*max_value/10, 8*max_value/10, 10*max_value/10])  # Điều chỉnh các giá trị trục theo nhu cầu

# Thêm chú thích
ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))

# Hiển thị biểu đồ
plt.show()
```

ví dụ với input:

Player 1: Levi Colwill

Player 2: Marc Cucurella

Attributes: ['xg', 'npxg', 'xg_assist', 'assists_per90', 'xg_assist_per90', 'xg_xg_assist_per90', 'npxg_xg_assist_per90', 'shots_per90', 'shots_on_target_per90', 'npxg_per_shot']


Player Attribute Comparison
assists_per90
xg_assist
npxg
xg
npxg_per_shot
shots_on_target_per90
shots_per90
npxg_xg_assist_per90
xg_xg_assist_per90
xg_assist_per90
0.26
0.52
0.78
1.04
1.30
Levi Colwill
Marc Cucurella