HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP

Môn: Lập trình Python

Giảng viên:	**Kim Ngọc Bách**
Lớp:	**INT13162-20241-11**
Họ và tên:	**Trần Thế Hưng**
Mã sinh viên:	**B22DCKH059**
Email:	**Tranthehung0511no3@gmail**

Hà Nội – 2024

1. Viết chương trình Python thu thập dữ liệu phân tích cầu thủ thi đấu hơn 90 phút của giải ngoại hạng Anh 2023 – 2024

- B1: Xác định đường dẫn chứa dữ liệu:
  + Base url: https://fbref.com
  + Đường dẫn đến trang chứa bảng các đội: /en/comps/9/2023-2024/2023-2024-Premier-League-Stats
- B2: Phân tích cấu trúc html:
  + Ta thấy trang ở B1 có nội dung gồm nhiều bảng tổng hợp chỉ số của 20 câu lạc bộ ngoại hạng Anh, mỗi hàng tương ứng với 1 đội, có kèm 1 thẻ a dẫn tới trang thống kê chi tiết của từng cầu thủ trong đội.
  + Với mỗi trang đã lấy được từ thẻ a, nội dung của nó bao gồm nhiều bảng với các thông tin cần lấy nằm ở các cột.
  + Đặc điểm chung của các cột này là thuộc tính "data-stat", ta sẽ dựa vào đây để lấy chính xác thông tin.
- B3: Sử dụng thư viện requests để lấy cấu trúc html và Beautifulsoup4 để phân tích cấu trúc html
  + Thư viện Beautifulsoup4 giúp phân tích cú pháp html dựa theo id, class, tag, thuộc tính thẻ và lấy các thông tin như thuộc tính, text thông qua các phương thức chọn (select).
  + Đầu tiên, gửi yêu cầu đến trang /en/comps/9/2023-2024/2023-2024-Premier-League-Stats, chọn 1 bảng có đầy đủ 20 thẻ a dẫn đến các câu lạc bộ.

```
response = requests.get(BASE_URL + EPL_23_24_PATH)
document = bs4.BeautifulSoup(response.text, 'html.parser')
```

  + Đối với mỗi câu lạc bộ, lấy thông tin từ các bảng và lưu vào cấu trúc dữ liệu ở dạng key – value như dictionary:

```
def fetch(url):
    res = requests.get(url)
    return bs4.BeautifulSoup(res.text, 'html.parser')

for linkElement in document.select('#stats_squads_keeper_for tbody tr a'):
        link = linkElement['href']
        page = fetch(BASE_URL + link)
        standard_stats = page.select('#stats_standard_9 tbody tr')
        keeper_table = page.select_one('#stats_keeper_9 tbody')
        advanced_GK_table = page.select_one('#stats_keeper_adv_9')
        passing_table = page.select_one('#stats_passing_9 tbody')
        gca_table = page.select_one('#stats_gca_9 tbody')
        def_actions = page.select_one('#stats_defense_9 tbody')
        possesion_table = page.select_one('#stats_possession_9 tbody')
```

```
        playing_time = page.select_one('#stats_playing_time_9 tbody')
        miscellaneous_table = page.select_one('#stats_misc_9 tbody')
      #...........
```

+ Thông tin của từng chỉ số được lấy thông qua phương thức get_text

```
      container.select_one(selector).get_text()
      #sử dụng CSS Selector
```

+ Mỗi cầu thủ được lưu vào 1 mảng và mảng được sắp xếp theo yêu cầu.

- B4: Kiểm tra dữ liệu:

  + Để đảm bảo dữ liệu được thu thập đúng cách, cần chọn ra 1 vài điểm dữ liệu để so sánh với dữ liệu gốc.

  + Lưu dữ liệu dưới dạng json để dễ dàng trực quan hơn.

```
with open('data/players.json', 'w', encoding='utf-8') as file:
        json.dump(players, file, ensure_ascii=False, indent=4)
```

- B5: Lưu dữ liệu dưới dạng csv

  + Sau khi đã xác nhận dữ liệu thu thập đúng, lưu lại dữ liệu dưới dạng csv

2. Phân tích dữ liệu

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

  + B1: Đọc dữ liệu từ file csv và xử lý các trường ‘NA’

```
df = pd.read_csv('data/results.csv')
df = df.replace('N/A', pd.NA)
```

  + B2: Lưu top 3 của mỗi trường với hàm nlargest (nsmallest) của thư viện pandas

```
top_3 = {}

for field in numericFields:
top_3[field] = [
  {
      "id": df['id'][index],
      "name": df['name'][index],
      "position": df['position'][index],
      "team": df['team'][index],
      "value": value
  }
  for index, value in df[field].dropna().nlargest(3).items()
]
```

  + B3: Lưu kết quả vào file csv

```
with open('data/top3.json', mode='w', encoding='utf-8') as file:
        json.dump(top_3, file, ensure_ascii=False, indent=4)
```

- Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội:
  + Thư viện pandas cung cấp các hàm median dùng để tính trung vi, mean dùng để tính trung bình và std dùng để tính độ lệch chuẩn.

```
def find(team, index):
        filtered_player = df[df['team'].str.contains(team)]
        yield index
        yield team or "all"
        for field in numericFields:
            yield filtered_player[field].dropna().median()
            yield filtered_player[field].dropna().mean()
            yield filtered_player[field].dropna().std(ddof=0)
```

  Với numericFields là các trường kiểu số, với các trường kiểu str (tên, quốc tịch) thì không tính. Hàm dropna() để loại bỏ các giá trị rỗng
  + Lưu các giá trị tính được vào file csv

```
with open('data/results2.csv', mode='w', encoding='utf-8',
newline='') as file:
        dataResults2.to_csv(file, index=False, encoding='utf-8')
```

- Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.
  + Biểu đồ histogram còn gọi là biểu đồ tần suất, là đồ thị thống kê biểu diễn sự phân bố của một tập dữ liệu liên tục thông qua các cột được vẽ, mỗi cột biểu diễn một loại hoặc khoảng lớp cụ thể. Chiều cao của thanh phản ánh tần suất hoặc số lượng điểm dữ liệu trong mỗi nhóm.
  + Thư viện matplotlib cung cấp hàm vẽ biểu đồ histogram

```
plt.hist(x, bins)
```

  + Với "bins" là các khoảng hoặc dải giá trị mà dữ liệu được chia thành. Chúng đại diện cho các nhóm giá trị dữ liệu liên tiếp và được sử dụng để hiển thị tần suất (frequency) của dữ liệu trong từng nhóm.
  + Số lượng bins: Quy định số lượng các khoảng mà dữ liệu được chia. Số lượng bins quá ít sẽ làm mất đi các chi tiết quan trọng, trong khi quá nhiều bins có thể tạo ra một biểu đồ phức tạp và khó hiểu.
  + Có nhiều phương pháp chọn bins, trong bài này phương pháp Sturges (cho trường hợp số lượng điểm rất nhỏ) và cải tiến thành Doane's Formula.

  Công thức Sturges:

$$bins = 1 + \log_2 n$$

  Công thức Doane:

$$bins = 1 + \log_2 n + \log_2\left(1 + \frac{|g1|}{\sqrt{variance\ of\ g1}}\right)$$

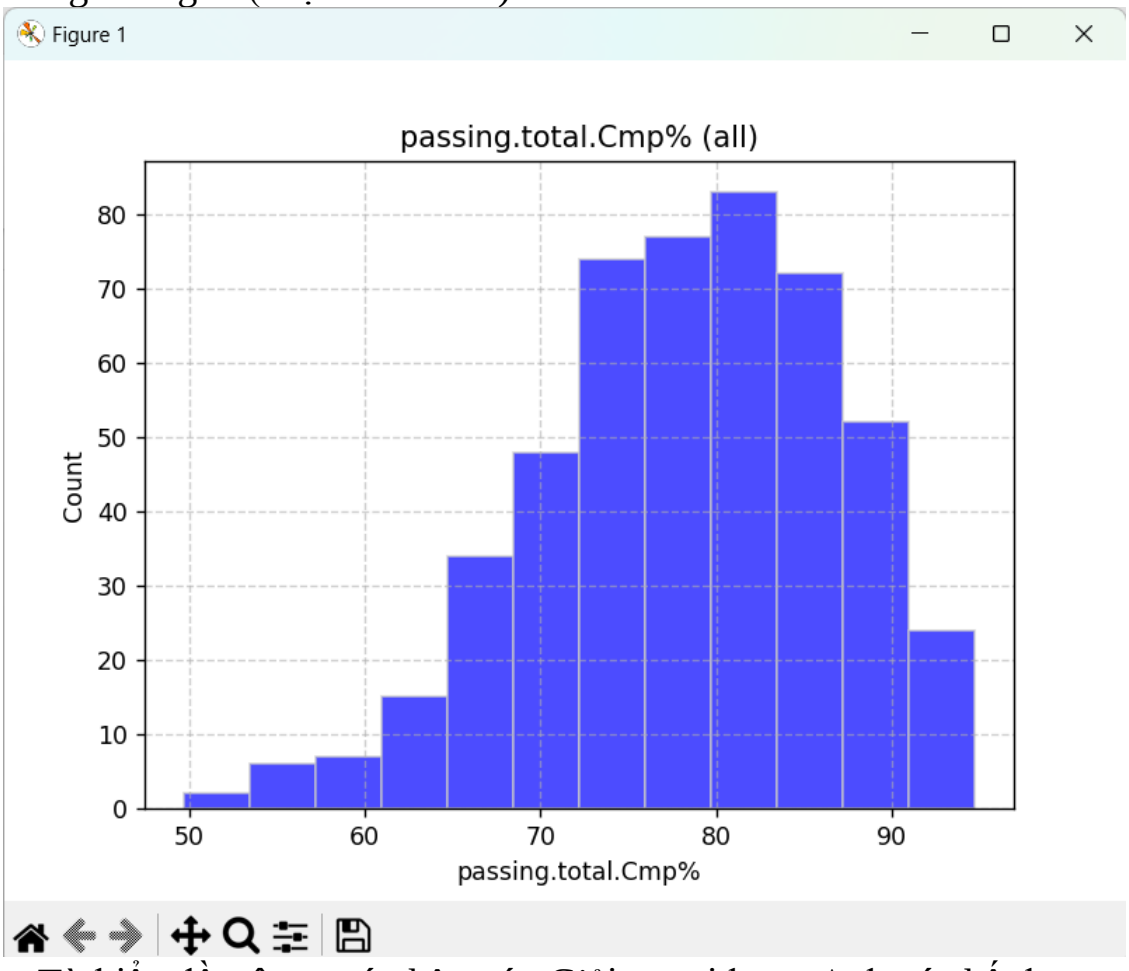
Với g1 là hệ số skewness(độ lệch chuẩn hóa của moment bậc 3) được tính bởi:

$$g1 = \frac{moment\ bậc\ 3}{(moment\ bậc\ 2)^{3/2}}$$

Moment bậc 3: Đo mức độ bất đối xứng của phân phối.
Moment bậc 2: Phương sai (σ^2)

$$Variance\ of\ g1 = \frac{6(n-2)}{(n+1)(n+3)}$$

+ Ví dụ vẽ histogram tỉ lệ đường chuyền thành công của các cầu thủ trong toàn giải (chọn bins = 12)



+ Từ biểu đồ trên ta có nhận xét: Giải ngoại hạng Anh có chất lượng cầu thủ tốt với tỉ lệ đường chuyền thành công nằm trong khoảng 80 – 83% chiếm đa số, chỉ có khoảng 24 cầu thủ có tỉ lệ đường chuyền chính xác trên 90%.

- Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số.
+ B1: Lấy dữ liệu của từng đội bóng với requests và bs4 (không sử dụng dữ liệu đã có vì dữ liệu này chứa các giá trị NA và thiếu 1 số cầu thủ chơi dưới 90 phút). Cách lấy tương tự bài 1.

+ B2: Sử dụng hàm max lên từng chỉ số để lấy đội bóng có chỉ số cao nhất

```
for stat in df.select_dtypes(include=['int64', 'float64']).keys():
    highest = df[df[stat] == df[stat].max()]
    print(f'Highest {stat}:')
    for index, club in highest.iterrows():
        print(f'  - {club["name"]} = {club[stat]}')
    print()
```

Kết quả:

```
Highest wins:
  - Manchester City = 28
  - Arsenal = 28

Highest draws:
  - Brighton = 12

Highest losses:
  - Sheffield Utd = 28

Highest goals:
  - Manchester City = 96

Highest goalsAgainst:
  - Sheffield Utd = 104

Highest goalDiff:
  - Manchester City = 62
  - Arsenal = 62

Highest points:
  - Manchester City = 91

Highest Pts/MP:
  - Manchester City = 2.39

Highest xG:
  - Liverpool = 87.8

Highest xGA:
  - Luton Town = 78.0

Highest xGD:
  - Arsenal = 48.2

Highest xGD/90:
  - Arsenal = 1.27
```

- Tìm đội bóng có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024

  + Phương pháp đánh giá phong độ: Sử dụng 1 công thức tổng hợp dựa trên các chỉ số thống kê, mỗi chỉ số có 1 hệ số riêng phụ thuộc vào tầm quan trọng của loại chỉ số đó, số điểm tối đa cho mỗi đội là 100 và tối thiểu là 0, đội có điểm cao hơn thì có phong độ tốt hơn.

  + Công thức: PV = 0.5RP + 0.3ODP + 0.2EP

  RP (Real performance) chiếm tỉ lệ cao nhất vì nó sử dụng kết quả bảng xếp hạng để tính. Với

  $$RP = \frac{43 * \log_{10}\big((20 - index) * 0.5 + 0.5\big)}{\log_{10} 10.5} + \frac{points}{2}$$

  index là vị trí của các đội tính từ 0 (đội vô địch có index = 0)

  công thức này đã được căn chỉnh nhiều lần sao cho RP luôn nằm trong [0, 100].

  Nếu 1 đội thắng cả 38 trận, đội này sẽ có RP = 100, ngược lại, nếu thua toàn bộ, đội này sẽ có RP = 0

  ODP (Offensive Deffensive Performance) chiếm tỉ lệ ít hơn RP do không hoàn toàn quyết định đến thứ hạng

  $$ODP_0 = goals - 0.75 * goalsAgainst$$

  $$ODP = \frac{ODP_0 - ODP_{0min}}{ODP_{0max} - ODP_{0min}} * 100$$

  goalAgainst (số bàn thua phải nhận) có hệ số thấp hơn số bàn thắng do đội mạnh nên có xu hướng dâng cao tấn công)

  EP (Expected Performance – phong độ kì vọng) quan trọng nhưng không phản ánh kết quả thực tế.

  $$EP_0 = xG - 0.75 * xGA$$

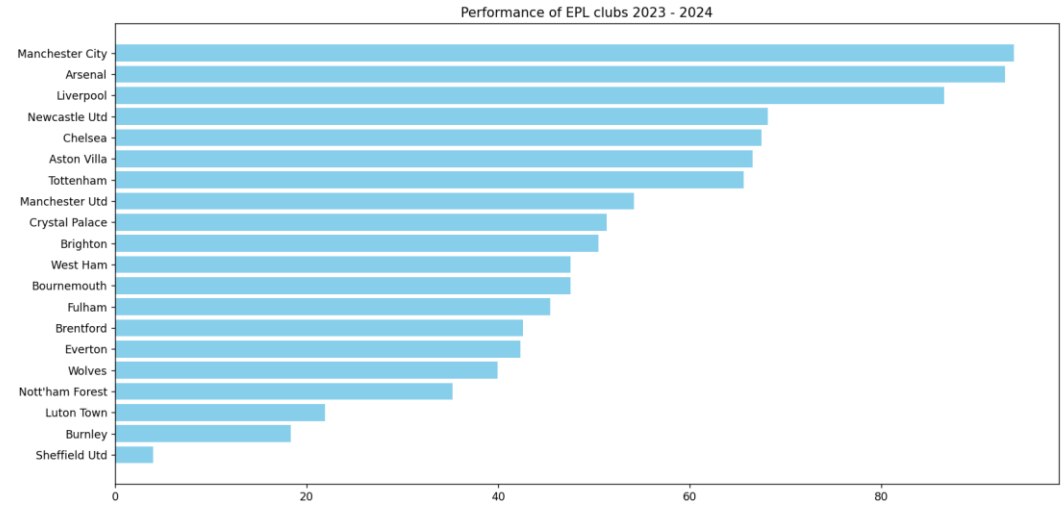
  $$EP = \frac{EP_0 - EP_{0min}}{EP_{0max} - EP_{0min}} * 100$$

  Tương tự ODP, xGA (số bàn thua kì vọng) có hệ số thấp hơn số bàn thắng kì vọng

  + Sau khi tính toán PV (Performance Value), đội nào có PV cao nhất thì có phong độ tốt nhất, có thể trực quan hóa bằng đồ thị

```
df['RP'] = 43 * np.log10(((20 - df.index) * 0.5 + 0.5)) /
np.log10(10.5) + df['points'] / 2

df['ODP'] =  df['goals'] - 0.75 * df['goalsAgainst']
df['ODP'] = (df['ODP'] - np.min(df['ODP'])) / (np.max(df['ODP']) -
np.min(df['ODP'])) * 100

df['EP'] = df['xG'] - 0.75 * df['xGA']
df['EP'] = (df['EP'] - np.min(df['EP'])) / (np.max(df['EP']) -
np.min(df['EP'])) * 100
```

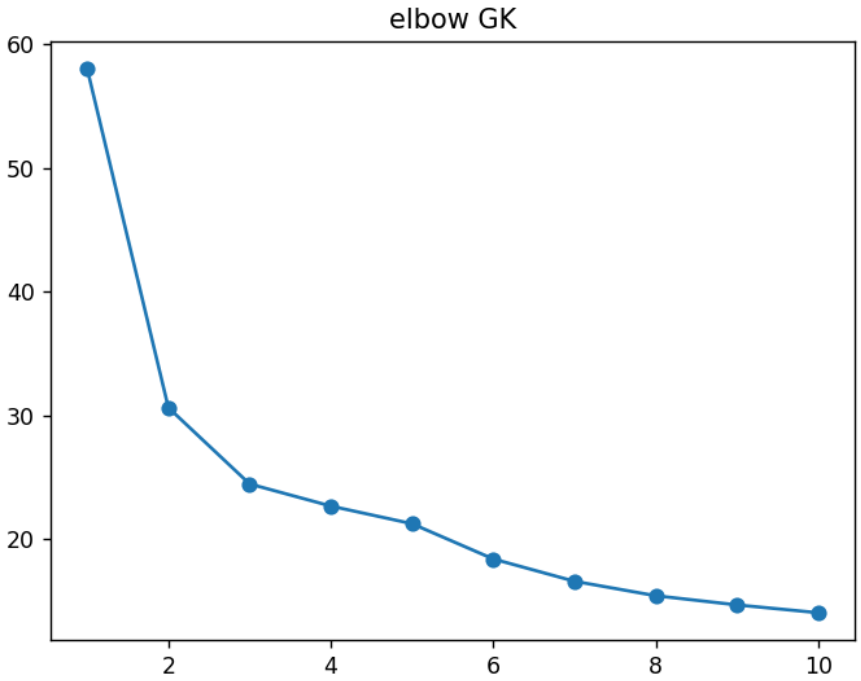
```
df['PV'] = 0.5 * df['RP'] + 0.3 * df['ODP'] + 0.2 * df['EP']

df = df.sort_values(by='PV', ascending=False)
print(df['PV'])

plt.barh(df['name'], df['PV'], color='skyblue')
plt.title("Performance of EPL clubs 2023 - 2024")
plt.gca().invert_yaxis()
plt.savefig("visualization/hist_club_performances.png")
plt.show()
```



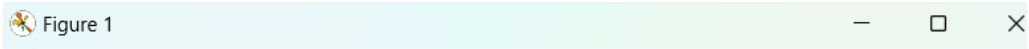
Theo đó, đội có phong độ tốt nhất là Manchester City

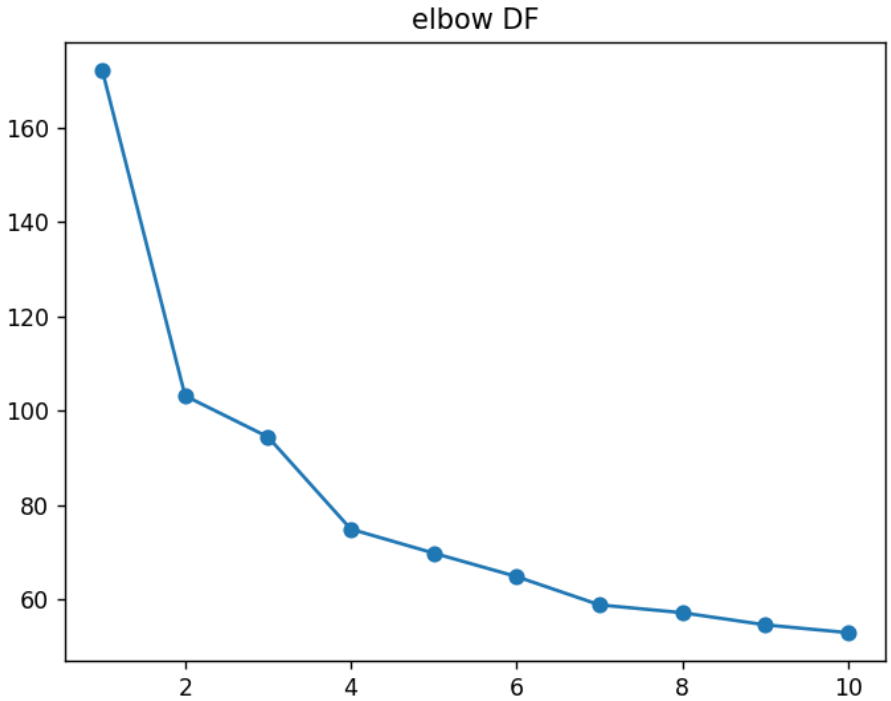
3. Phân tích nâng cao

- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau:
  + B1: Chia cầu thủ theo các vị trí: thủ môn, phòng ngự, tiền vệ, tiền đạo.
  + B2: Đối với mỗi vị trí, chọn ra các chỉ số mà được đánh giá là ảnh hưởng nhiều đến lối chơi.
  + B3: Chọn số cụm, trong bài này sử dụng phương pháp elbow

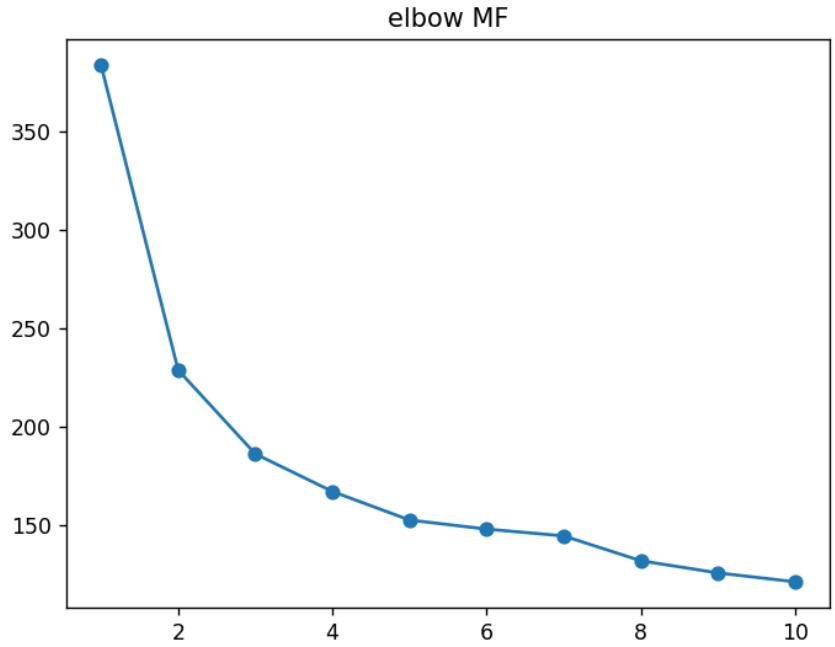


Chọn số cụm là 3 đối với thủ môn

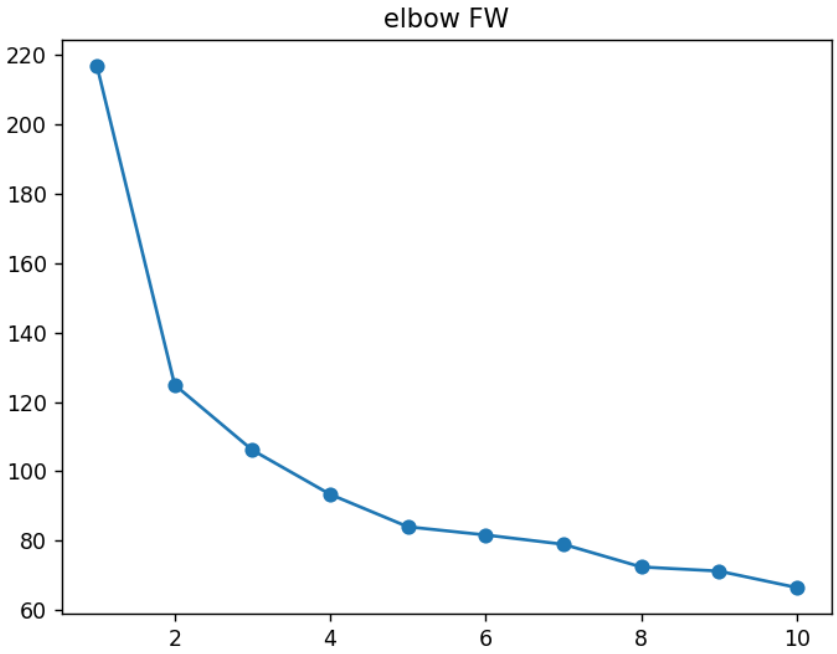




Chọn số cụm là 4 đối với cầu thủ phòng ngự



Chọn số cụm là 4 đối với tiền vệ



Chọn số cụm là 4 đối với tiền đạo

+ B4: Sau khi đã chọn được số cụm, đưa vào mô hình K-means để huấn luyện, sau đó lưu dữ liệu đã được chia vào file json (hoặc csv)

```
def clustering(position, features: list, k = 3):
    scaler = MinMaxScaler()

    clusData = df[df['position'].str.contains(position)][['id', 'name', 'team'] +
features].reset_index()
    clusData = clusData.apply(lambda col: col.fillna(col.min()))
    clusDataScaled = scaler.fit_transform(clusData[features])
```

```
    # elbowTest(clusDataScaled, position)

    model = KMeans(n_clusters = k, random_state = 42)
    clusData['cluster'] = model.fit_predict(clusDataScaled)

    groups = clusData.groupby('cluster').agg(list).reset_index()
    output = {}
    for index, row in groups.iterrows():
        output[f'cluster-{index}'] = {
            "players": [f'{name} - {id} - {team}' for name, id, team in
zip(row["name"], row['id'], row['team'])],
        }
        for feature in features:
            output[f'cluster-{index}'][feature] = np.array(row[feature]).mean()

    return output

if __name__ == '__main__':
    results = {}
    results['GK'] = clustering('GK', GK_FEATURES, 3)
    results['DF'] = clustering('DF', DF_FEATURES, 4)
    results['MF'] = clustering('MF', MF_FEATURES, 4)
    results['FW'] = clustering('FW', FW_FEATURES, 4)

    with open("data/clusters.json", mode='w', encoding='utf-8') as file:
        json.dump(results, file, ensure_ascii=False, indent=4)
```

- Nhận xét về kết quả:

+ Thủ môn: Gồm 3 cụm

1) Nhóm thủ môn dự bị hoặc thi đấu ít, chỉ số tổng thể thấp.
2) Nhóm thủ môn hàng đầu, hiệu quả toàn diện, thường xuất hiện trong các đội mạnh.
3) Nhóm thủ môn thi đấu nhiều nhất, chịu áp lực lớn hơn, nhưng vẫn có khả năng cứu thua tốt.

+ Cầu thủ phòng ngự: gồm 4 cụm:

1) Hậu vệ ít thi đấu, tập trung vào phòng ngự cơ bản.
2) Hậu vệ biên và trung vệ hỗ trợ tấn công.
3) Trung vệ tập trung vào phòng ngự ở khu vực cấm địa.
4) Hậu vệ toàn diện, thi đấu hiệu quả cả trong phòng ngự lẫn tổ chức tấn công.

+ Tiền vệ: gồm 4 cụm

1) Tiền vệ dự bị, đóng vai trò hỗ trợ.
2) Thiên về sáng tạo và tấn công.
3) Tiền vệ toàn diện giữa công và thủ.

4) Tiền vệ chủ lực, toàn diện và dẫn dắt lối chơi.

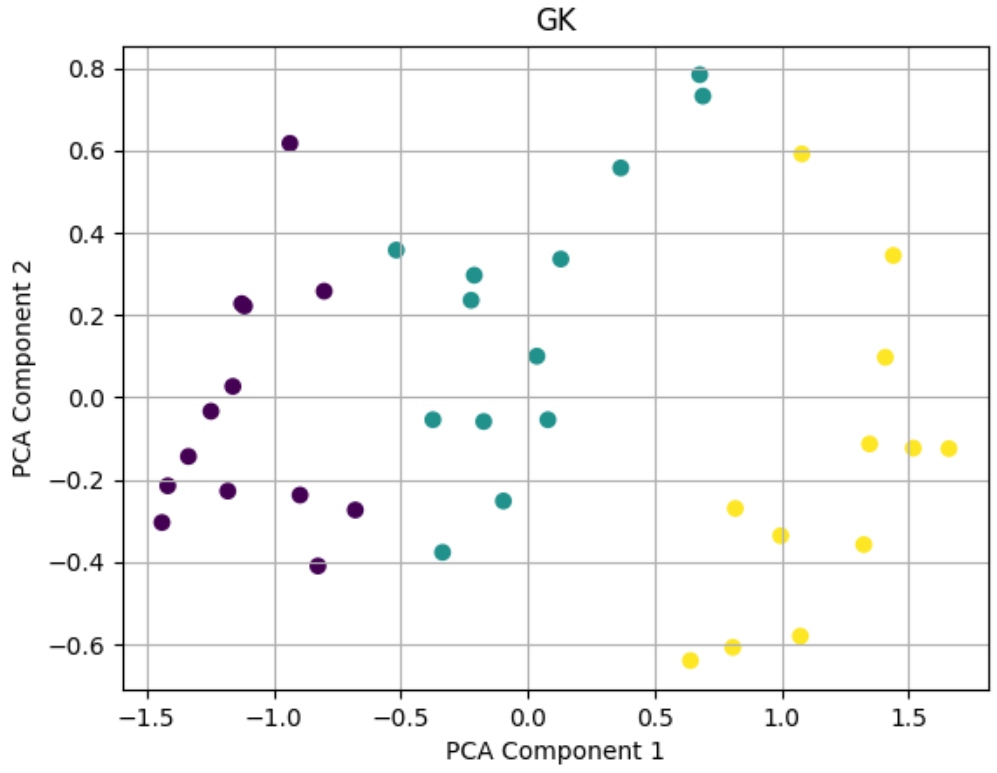
+ Tiền đạo: gồm 4 cụm:

1) Phù hợp cho cầu thủ dự bị hoặc có thời gian thi đấu hạn chế.
2) Tiền đạo ổn định, có khả năng tạo cơ hội và ghi bàn đáng tin cậy.
3) Tiền đạo ngôi sao, toàn diện với khả năng tấn công và chuyền bóng xuất sắc.
4) Tiền đạo hỗ trợ tốt, tham gia phòng ngự và có khả năng tạo bất ngờ từ sút xa.

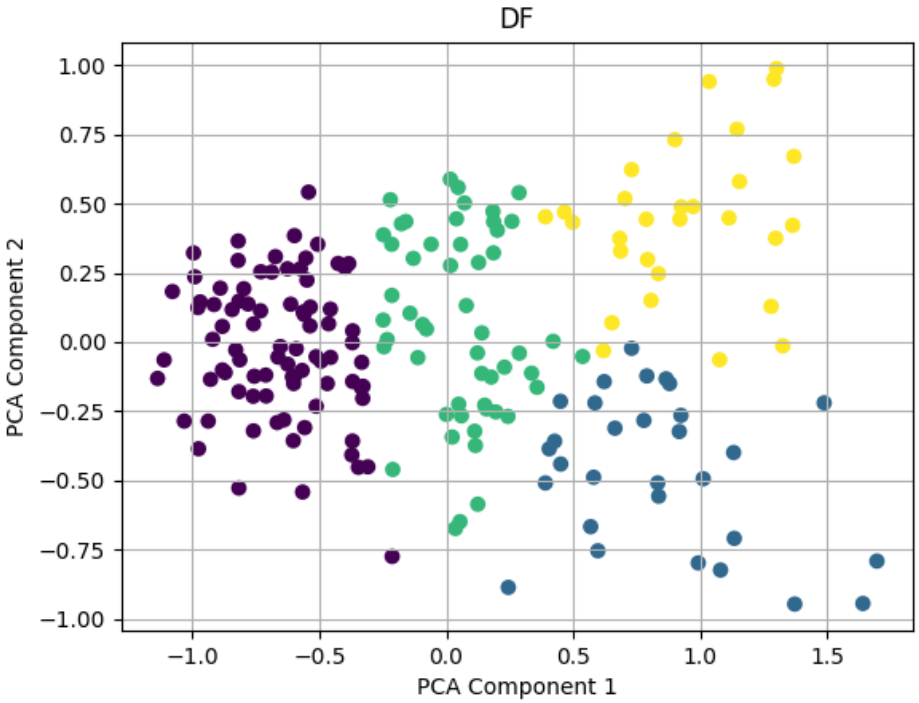
- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

```
def visualize(scaled, cluster, position):
    model = PCA(n_components=2)

    pca_result = model.fit_transform(scaled)

    plt.scatter(x=pca_result[:, 0], y=pca_result[:, 1], c=cluster)
    plt.title(position)
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.grid()
    plt.savefig(f"visualization/cluster_{position}.png")
    plt.show()
```

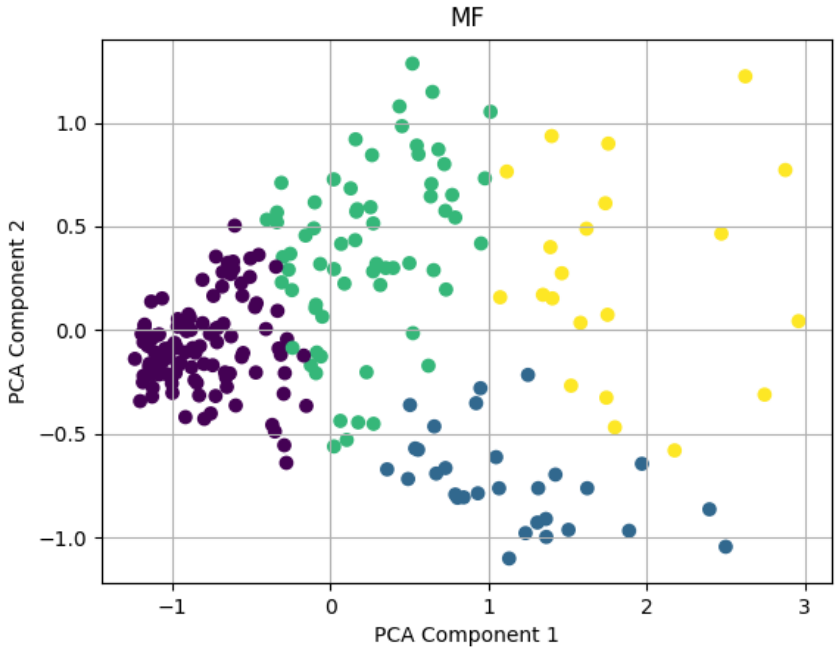
Thủ môn:



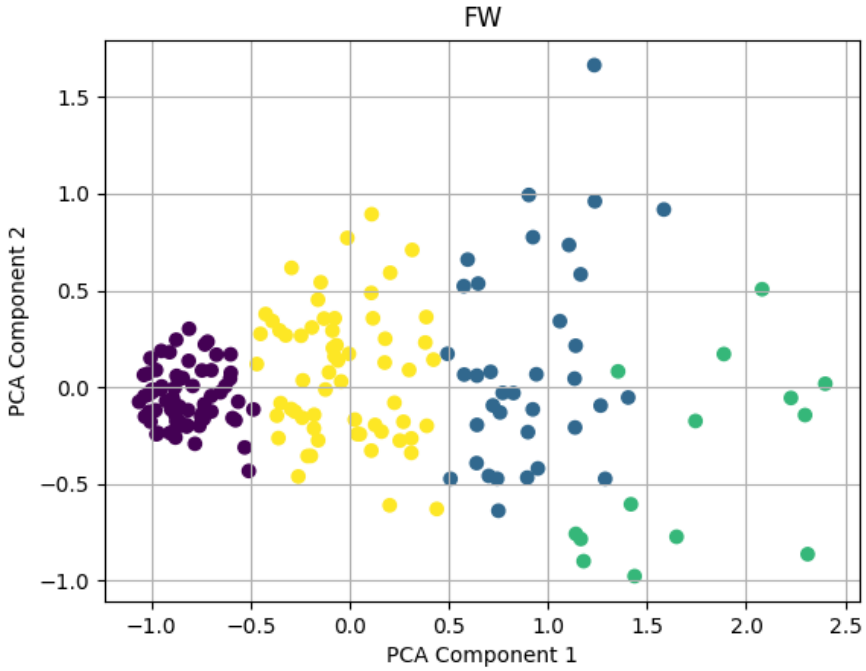
Cầu thủ phòng ngự:



Tiền vệ:



Tiền đạo:



- Chương trình so sánh 2 cầu thủ bằng radar chart:

  + Tính năng:

    1) Được chọn cầu thủ nếu nhập vào tên có nhiều cầu thủ trùng
    2) Nếu không nhập thuộc tính cần so sánh, chương trình tự động phát hiện vị trí chơi chung của 2 cầu thủ, qua đó chọn ra các chỉ số phù hợp.
    3) Nếu không nhập thuộc tính cần so sánh và 2 cầu thủ không chơi cùng vị trí, chương trình sẽ báo lỗi

  + Cách sử dụng:

    python radarChartPlot.py --p1 --p2 –Attribute

    Trong đó, p1 và p2 là tên cầu thủ, bắt buộc phải nhập

    Ví dụ:

```
python code/radarChartPlot.py --p1 "Rodri" --p2 "Declan Rice"
python code/radarChartPlot.py --p1 "Haaland" --p2 "Foden" --Attribute "playingTime.minutes, performance.goals, performance.assits, expected.npxG, passing.total.Cmp"
```

4. Thu thập giá chuyển nhượng cầu thủ

- Thu thập giá chuyển nhượng cầu thủ:
  + Địa chỉ: https://www.footballtransfers.com
  + Phân tích: Trang web không sử dụng SSR (Server side rendering) như bài 1 mà cung cấp API để truy vấn dữ liệu, url: https://www.footballtransfers.com/us/transfers/actions/confirmed/overview
  + API này được gọi với phương thức POST, body được truyền dưới dạng key – value
  + Một số trường quan trọng:
    1) season: id của mùa giải, mùa giải 2023 – 2024 có id là 5847
    2) page: Số thứ tự trang của API
    3) pageItems: Số lượng bản ghi trong trang (càng cao càng ít phải gọi API). Đối với trang web này, số lượng bản ghi tối đa có thể lấy là 4848 (Đã thử nhiều lần bằng công cụ Postman)
  + Dữ liệu thu được dưới dạng json
  + Gọi API với requests

```
API =
'https://www.footballtransfers.com/us/transfers/actions/confirmed/overview'


MAX_ITEMS_PER_REQ = 4848
MAX_ITEMS = 13353
SEASON_ID = 5847 # 2023 - 2024

data = []
```

```
for page in range(1,math.ceil(MAX_ITEMS / MAX_ITEMS_PER_REQ)+1):
    body = {
        "season": 5847,
        "page": page,
        "pageItems": MAX_ITEMS_PER_REQ
    }

    response = requests.post(API, data=body)
    print(response)

    rawData = json.loads(response.text)
    for record in rawData['records']:
        data.append({
            "player_id": record['player_id'],
            "player_name": record['player_name'],
            "country_name": record['country_name'],
            "age": record['age'],
            "position_name": record['position_name'],
            "club_from_name": record['club_from_name'],
            "club_to_name": record['club_to_name'],
            "amount": record['amount'],
            "date_transfer": record['date_transfer']
        })
```

+ Lưu dữ liệu:

```
with open(JSON_DEST, mode='w', encoding='utf-8') as jsonFile:
    json.dump(data, jsonFile, ensure_ascii=False, indent=4)

with open(CSV_DEST, mode='w', newline='', encoding='utf-8') as
csvFile:
    fieldnames = data[0].keys()
    writer = csv.DictWriter(csvFile, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerows(data)
```

- Đề xuất phương pháp định giá cầu thủ

+ Định giá dựa trên dữ liệu thống kê: Sử dụng các thông số thống kê từ trận đấu (bàn thắng, kiến tạo, xG, số phút thi đấu, …), Tạo trọng số cho từng chỉ số dựa trên vai trò cầu thủ (VD: Tiền đạo ưu tiên bàn thắng/xG, Hậu vệ ưu tiên tranh chấp/phòng ngự).

Ví dụ:

$$Performance\ Score = w1 * goals + w2 * assists + \cdots$$

Với w1, w2, w3 là trọng số

+ Phân tích tuổi và giá trị tiềm năng: Sử dụng mô hình hồi quy tuyến tính để ước lượng giá trị cầu thủ theo tuổi tác.

+ So sánh thị trường: So sánh cầu thủ với những người có thống kê tương tự đã được chuyển nhượng gần đây để đưa ra mức giá hợp lý.