HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

****

BÁO CÁO BÀI TẬP LẬP TRÌNH
MÔN PYTHON

Giảng viên hướng dẫn	:Kim Ngọc Bách
Nhóm	:11
Họ tên	:Nguyễn Trung Kiên
Mã sinh viên	:B22DCKH061

Hà Nội-2024

I. Thu thập dữ liệu và tiền xử lý

. Định Nghĩa Hàm

Hàm geturl(url): Hàm này thực hiện yêu cầu HTTP tới URL được cung cấp và trả về nội dung HTML dưới dạng đối tượng BeautifulSoup để dễ dàng phân tích cú pháp.

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

Hàm extract_table_data(page, table_id, data_append_csv, start_col, end_col, default_count): Hàm này trích xuất dữ liệu từ một bảng cụ thể trong trang HTML. Nếu bảng hoặc dữ liệu không được tìm thấy, nó sẽ trả về một danh sách các giá trị "N/A".

```
def extract_table_data(page, table_id, data_append_csv, start_col, end_col, default_count):

    table_data = []
    try:
        table = page.find("table", {"id": table_id})
        if table:
            for row in table.tbody.find_all('tr'):
                cols = row.find_all("td")
                if data_append_csv == row.find("th", {"data-stat": "player"}).get('data-append-csv'):
                    for col in cols[start_col:end_col]:
                        table_data.append(col.get_text() if col.get_text() != "" else "N/A")
                    return table_data
    except:
        pass
    # Nếu không có dữ liệu hợp lệ, trả về danh sách "N/A"
    return ["N/A"] * default_count
```

Thiết Lập URL Cơ Sở và Điểm Cuối

Bạn cần xác định URL cơ sở của trang web và điểm cuối cụ thể cho thống kê Premier League 2023-2024:

```
base_url = "https://fbref.com"
epl_2023_2024 = "/en/comps/9/2023-2024/2023-2024-Premier-League-Stats"
```

Lấy và Phân Tích Trang Chính

thực hiện yêu cầu HTTP để lấy trang chính chứa thống kê Premier League và phân tích cú pháp nội dung HTML của trang này:

```
response = requests.get(base_url + epl_2023_2024)
document = BeautifulSoup(response.text, "html.parser")
```

Lặp Qua Các Liên Kết Cầu Thủ

Bạn sẽ lặp qua từng liên kết cầu thủ được tìm thấy trong trang chính, lấy và phân tích từng trang cá nhân của cầu thủ:

```
for linkal in document.select('#stats_squads_keeper_for tbody tr a'):
    link = linkal['href']
    page = geturl(base_url + link)
    team = linkal.get_text()
    try:
        table = page.find("table", {"id": "stats_standard_9"})
        for row in table.tbody.find_all('tr'):
```

Tạo và Sắp Xếp DataFrame

Sau khi thu thập dữ liệu, bạn chuyển đổi danh sách players thành một DataFrame của pandas, thêm một cột tạm thời để sắp xếp cầu thủ theo từ đầu tiên của tên và tuổi, sau đó xóa cột tạm thời này:

```
dataFrame['first_word'] = dataFrame["Name"].str.split().str[0]
dataFrame = dataFrame.sort_values(by=['first_word', 'Age'], ascending=[True, False])
dataFrame.drop(columns=['first_word'], inplace=True)
```

II. Phân tích thống kê

1.Cầu thủ hàng đầu theo chỉ số:

- **Tìm ra 3 cầu thủ có chỉ số cao nhất và thấp nhất cho mỗi chỉ số:**

-Đọc dữ liệu từ file csv:

```
df = pd.read_csv('result.csv')
```

-Định Nghĩa Các Thuộc Tính

Xác định danh sách các thuộc tính mà bạn muốn tìm top 3 cao nhất và thấp nhất:

Attributes : **PlayingTime.MatchesPlayed", "PlayingTime.Starts"….**

Tìm Top 3 Cao Nhất và Thấp Nhất cho Mỗi Chỉ Số

Bạn sẽ tìm top 3 cao nhất và thấp nhất cho mỗi chỉ số trong danh sách Attributes:

```
top_bottom = {}
for attr in attributes:
    top_bottom[attr] = {
        'Top 3': df.nlargest(3, attr)[['Name', attr]].values.tolist(),
        'Bottom 3': df.nsmallest(3, attr)[['Name', attr]].values.tolist()
    }
```

Cuối cùng là in ra kết quả top 3 cao nhất và thấp nhất cho mỗi chỉ số.

2.Tính trung vị, giá trị trung bình và độ lệch chuẩn cho mỗi chỉ số, cả trên toàn bộ giải đấu và theo từng đội bóng.

Đầu tiên cần đọc dữ liệu từ file csv

-Định Nghĩa Các Thuộc Tính

-Xác định danh sách các thuộc tính mà bạn muốn tính toán các giá trị trung vị (median), trung bình (mean) và độ lệch chuẩn (std):

attributes :"PlayingTime.MatchesPlayed", "PlayingTime.Starts", "PlayingTime.Minutes"…

- Tạo một DataFrame để lưu kết quả tính toán cho toàn bộ giải đấu và từng đội:

```
results = pd.DataFrame(columns=[''] + [f"{stat} of {attr}" for attr in attributes for stat in ['Median', 'Mean', 'Std']])
```

- Tính Toán cho Toàn Bộ Giải Đấu

Tính toán các giá trị trung vị, trung bình và độ lệch chuẩn cho toàn bộ giải đấu:

```
all_stats = df[attributes].agg(['median', 'mean', 'std']).T.values.flatten()
results.loc[0] = ['all'] + all_stats.tolist()
```

- Tính Toán cho Từng Đội

Lặp qua từng đội và tính toán các giá trị trung vị, trung bình và độ lệch chuẩn cho từng đội:

```
team_stats_list = []
for team in df['Team'].unique():
    team_df = df.loc[df['Team'] == team]
    team_stats = team_df[attributes].agg(['median', 'mean', 'std']).T.values.flatten()
    team_stats_list.append([team] + team_stats.tolist())
results = pd.concat([results, pd.DataFrame(team_stats_list, columns=results.columns)], ignore_index=True)
```

Cuối cùng, lưu kết quả vào một tệp CSV có tên results2.csv.

3.Biểu đồ tần số (Histogram):

- Tạo biểu đồ tần số cho sự phân bố của mỗi chỉ số trên tất cả các cầu thủ và theo từng đội, sử dụng thư viện matplotlib.

-đọc dữ liệu từ tệp CSV đã lưu

Xác định danh sách các thuộc tính mà bạn muốn vẽ biểu đồ phân bố:

attributes = ["PlayingTime.MatchesPlayed"]

- Vẽ Histogram Phân Bố của Mỗi Chỉ Số của Các Cầu Thủ trong Toàn Giải

Lặp qua từng thuộc tính trong danh sách attributes và vẽ biểu đồ histogram phân bố của mỗi chỉ số cho tất cả các cầu thủ:

```
for attr in attributes:
    plot.figure(figsize=(10, 6))
    df[attr].hist(bins=30, edgecolor='black')
    plot.title(f'Histogram of {attr} for all players')
    plot.xlabel(attr)
    plot.ylabel('Frequency')
    plot.savefig(f'histograms/{attr}_all_players.png')
    plot.close()
```

-Vẽ Histogram Phân Bố của Mỗi Chỉ Số của Các Cầu Thủ trong Mỗi Đội Lặp qua từng đội và từng thuộc tính trong danh sách attributes, sau đó vẽ biểu đồ histogram phân bố của mỗi chỉ số cho các cầu thủ trong từng đội:

```
for team in df['Team'].unique():
    team_df = df[df['Team'] == team]

    for attr in attributes:
        plot.figure(figsize=(10, 6))
        team_df[attr].hist(bins=30, edgecolor='black')
        plot.title(f'Histogram of {attr} for {team}')
        plot.xlabel(attr)
        plot.ylabel('Frequency')
        plot.savefig(f'histograms/{attr}_{team}.png')
        plot.close()
```

4.Phân tích đội bóng có phong độ tốt nhất:

- **Tìm đội bóng có phong độ tốt nhất theo từng chỉ số bằng cách tổng hợp các chỉ số của cầu thủ và đánh giá điểm số tổng quát của từng đội.**

1. Mục tiêu

- Xác định đội bóng có chỉ số tốt nhất trong từng chỉ số và đội có phong độ tốt nhất tổng thể, dựa trên các chỉ số cầu thủ đã cho.

2. Dữ liệu và Chỉ số

- Dữ liệu được lấy từ file results.csv, chứa thông tin cầu thủ và đội bóng theo từng chỉ số.
- Các chỉ số (attributes) bao gồm các khía cạnh như: thời gian thi đấu, hiệu suất bàn thắng, cản phá, kiến tạo, và các thông số liên quan đến vị trí và kiểm soát bóng.

3. Các bước thực hiện

Bước 1: Đọc dữ liệu từ file CSV và xác định các chỉ số cần phân tích:

- Sử dụng pandas để đọc dữ liệu và xác định các chỉ số (attributes) mà bạn muốn phân tích.

Bước 2: Tìm đội bóng có chỉ số cao nhất ở từng chỉ số.

- Với mỗi chỉ số trong attributes, xác định đội bóng đứng đầu bằng cách tìm cầu thủ có giá trị chỉ số cao nhất. Tên đội bóng của cầu thủ đó sẽ là đội dẫn đầu cho chỉ số đó.
- Lưu thông tin đội bóng dẫn đầu từng chỉ số vào một từ điển (top_teams).

Bước 3: Đếm số lần đội bóng đứng đầu các chỉ số.

- Tạo một từ điển (phong_do) để ghi lại số lần một đội bóng đứng đầu các chỉ số. Điều này sẽ giúp xác định đội có phong độ tốt nhất tổng thể dựa trên tần suất dẫn đầu.

Bước 4: Tìm đội bóng có phong độ tốt nhất.

- Xác định đội có số lần dẫn đầu nhiều nhất, lưu kết quả này vào một bảng để tổng hợp.

Bước 5: Tạo bảng tổng hợp kết quả và lưu vào file CSV.

- Tạo bảng top_teams_df chứa các đội dẫn đầu từng chỉ số và bảng best_performance_df cho đội có phong độ tốt nhất tổng thể.
- Kết hợp hai bảng trên thành bảng cuối cùng và lưu vào file results_Chiso-Phongdo.csv.

4. Kết quả

- File results_Chiso-Phongdo.csv sẽ chứa thông tin về đội dẫn đầu từng chỉ số và đội có phong độ tổng thể tốt nhất, dễ dàng phân tích trong các phần sau của báo cáo.
    -

III. Phân cụm và Giảm chiều dữ liệu

1. Phân cụm K-Means:

- Sử dụng thuật toán K-Means để phân loại các cầu thủ thành các nhóm có đặc điểm chỉ số tương tự.
- Sử dụng phương pháp Elbow để xác định số nhóm tối ưu và phân tích kết quả phân cụm để phân loại loại hình cầu thủ.

-đọc dữ liệu từ tệp CSV đã lưu

- Xác định danh sách các thuộc tính mà bạn muốn sử dụng để phân cụm các cầu thủ:

attributes = "PlayingTime.MatchesPlayed", "PlayingTime.Starts"….

- Xử Lý Dữ Liệu

Điền các giá trị thiếu bằng giá trị trung bình của các thuộc tính tương ứng và chuẩn hóa dữ liệu:

```
X = df[attributes].fillna(df[attributes].mean())

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

-Phân Cụm Dữ Liệu

Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau:

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X_scaled)

# Gán nhãn nhóm cho từng cầu thủ
df['Cluster'] = kmeans.labels_
```

-Vẽ Biểu Đồ Phân Tán của Các Nhóm Cầu Thủ
Vẽ biểu đồ phân tán của các nhóm cầu thủ dựa trên hai thuộc tính đầu tiên trong danh sách attributes:

```
plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans.labels_, cmap='viridis')
plt.xlabel(attributes[3])
plt.ylabel(attributes[5])
plt.title('K-means Clustering of Players')
plt.colorbar(label='Cluster')
plt.show()
```

✓ **Nhận xét về kết quả:**

- **Số lượng cụm đề xuất**: Khi dùng phương pháp Elbow, số lượng cụm phù hợp thường từ 4 đến 6 cụm cho dữ liệu cầu thủ vì:
  - Đủ để phân tách rõ các vai trò chính (như thủ môn, hậu vệ, tiền vệ, tiền đạo).
  - Cho phép nhóm các cầu thủ có phong cách chơi tương tự hoặc các chỉ số gần giống nhau, đặc biệt trong cùng một vị trí.
- **Chất lượng phân nhóm**:

- Nếu các cầu thủ trong mỗi nhóm có sự tương đồng cao về chỉ số (ví dụ: cụm thủ môn có nhiều chỉ số về "Saves" và "Save%"), điều này cho thấy cụm được hình thành tốt.
- Nếu có sự chồng chéo giữa các cụm (cầu thủ từ vai trò khác nhau trong cùng cụm), có thể điều chỉnh thêm số lượng cụm hoặc thêm một bước xử lý bổ sung để phân biệt rõ vai trò của cầu thủ.

2. PCA để giảm chiều dữ liệu:

- Sử dụng phương pháp PCA để giảm dữ liệu xuống 2 chiều để trực quan hóa.
- Vẽ biểu đồ phân cụm trên mặt phẳng 2D để thấy rõ cách các cầu thủ được nhóm lại dựa trên các chỉ số tương đồng.

- đọc dữ liệu từ tệp CSV đã lưu

- Xác định danh sách các thuộc tính mà bạn muốn sử dụng để phân tích thành phần chính (PCA)

-Xử Lý Dữ Liệu

Điền các giá trị thiếu bằng giá trị trung bình của các thuộc tính tương ứng và chuẩn hóa dữ liệu:

```
# Lấy dữ liệu các thuộc tính cần tính toán và điền giá trị trung bình cho các giá trị bị thiếu
X = df[attributes].fillna(df[attributes].mean())
# Chuẩn hóa dữ liệu
CHoa = StandardScaler()
X_CHoa = CHoa.fit_transform(X)
```

-Giảm Số Chiều Dữ Liệu bằng PCA

Sử dụng thuật toán PCA để giảm số chiều dữ liệu xuống 2 chiều:

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_CHoa)
```

-Vẽ Biểu Đồ Phân Cụm Các Điểm Dữ Liệu trên Mặt 2D:

```
plt.figure(figsize=(10, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c='blue', edgecolor='k', s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Player Attributes')
plt.show()
```

3. **Biểu đồ Radar so sánh cầu thủ:**

- Tạo hàm vẽ biểu đồ radar để so sánh hai cầu thủ dựa trên các chỉ số cụ thể.
- Triển khai các tham số dòng lệnh để nhập tên cầu thủ và các chỉ số cần so sánh, sau đó vẽ biểu đồ radar so sánh.

- Tạo Hàm **radar_factory**

Hàm này tạo một biểu đồ radar với số lượng trục được chỉ định và đăng ký nó như một loại hình chiếu mới, cho phép khung là 'hình tròn' hoặc 'hình đa giác'

-Tạo Hàm **plot_radar_chart**

Hàm này vẽ biểu đồ radar để so sánh hai cầu thủ dựa trên các thuộc tính đã chọn

- Phân Tích Các Đối Số Dòng Lệnh

Sử dụng argparse để phân tích các đối số dòng lệnh và gọi hàm plot_radar_chart với các đối số này.

IV. Thu thập dữ liệu và định giá chuyển nhượng

1. **Dữ liệu chuyển nhượng**:
   - Thu thập dữ liệu phí chuyển nhượng của cầu thủ từ footballtransfers.com trong mùa giải 2023-2024.

-Xác định URL API và đường dẫn tệp CSV để lưu dữ liệu:

```
API = os.getenv('FOOTBALL_API_URL', 'https://www.footballtransfers.com/us/transfers/actions/confirmed/overview')
csv_file_path = 'data/transfer.csv'
```

-Xác định số lượng mục tối đa trên mỗi yêu cầu, tổng số mục và ID mùa giải:

```
max_item_per_req = 4848
max_items = 13353
season_id = 5847 #id 2023-2024
```

-Xác định danh sách các câu lạc bộ mà bạn muốn thu thập dữ liệu:

```
clubs = ["Arsenal FC", "Aston Villa", "Bournemounth", "Brentford FC", "Brighton & Hove Albion", "Burnley FC", "Chelsea",
         "Crystal Palace", "Everton FC", "Fullham FC", "Ipswich Town", "Leicester", "Liverpool FC", "Luton Town",
         "Manchester City", "Manchester United", "Newcastle United", "Nottingham Forest", "Sheff Utd", "Southampton",
         "Tottenham", "Wes Ham United", "Wolves"]
```

-Lặp Qua Các Trang Để Thu Thập Dữ Liệu

Bạn sử dụng một vòng lặp for để lặp qua các trang của API. Số lượng trang được tính bằng cách chia tổng số mục (max_items) cho số lượng mục tối đa trên mỗi yêu cầu (max_item_per_req) và làm tròn lên ,Trong mỗi lần lặp, bạn tạo một từ điển body chứa các tham số cần thiết cho yêu cầu POST, bao gồm ID mùa giải (season_id), số trang hiện tại (page), và số lượng mục trên mỗi trang (pageItems):

```
for page in range(1, math.ceil(max_items / max_item_per_req) + 1):
    body = {
        "season": season_id,
        "page": page,
        "pageItems": max_item_per_req
    }
```

- lặp qua từng bản ghi trong rawData['records']. Nếu câu lạc bộ chuyển đi (club_from_name) hoặc câu lạc bộ chuyển đến (club_to_name) nằm trong danh

sách các câu lạc bộ (clubs), bạn lưu trữ thông tin của bản ghi đó vào danh sách data:

```
for record in rawData['records']:
    if record["club_from_name"] in clubs or record["club_to_name"] in clubs:
        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"],
        })
```