BÁO CÁO BÀI TẬP LỚN PYTHON - THẦY KIM NGỌC BÁCH

Họ và tên: Nguyễn Tiến Hưng
MSV: B22DCCN415
Nhóm: 11

Câu 1:

1. Import thư viện

+ BeautifulSoup từ bs4: Dùng để phân tích và trích xuất dữ liệu từ HTML.
+ Comment: Dùng để xử lý các bình luận HTML ẩn trong trang.
+ Requests: Dùng để gửi yêu cầu HTTP và nhận phản hồi từ một URL.
+ Pandas: Dùng để tạo và xử lý các DataFrame.

2. Hàm crawler

+ Mục đích: Tải về và xử lý bảng dữ liệu từ một URL nhất định, sau đó lưu vào tệp CSV.
+ Tham số:
  url: URL của trang web để crawl.
  id_div: ID của thẻ div chứa dữ liệu cần thiết.
  cnt: Số thứ tự của tệp CSV được lưu.
+ Quy trình:
  Gửi yêu cầu tới url và tạo đối tượng BeautifulSoup để phân tích HTML.
  Tìm thẻ div với id tương ứng và lấy các đoạn bình luận (Comment) chứa dữ liệu bảng.
+ Chuyển đoạn văn bản của bình luận thành các thẻ HTML để dễ xử lý.
+ Tạo từ điển ans1 để lưu dữ liệu, với các cột xác định từ thẻ <th>.
+ Lặp qua các hàng dữ liệu <tr> từ hàng thứ 2 trở đi để lấy dữ liệu từ các thẻ <td>, lưu vào từ điển.
+ Tạo DataFrame từ ans1 và lưu dưới dạng CSV với tên table{cnt}.csv.

3. Hàm clean_data

+ Mục đích: Kết hợp các bảng dữ liệu từ nhiều tệp CSV lại thành một DataFrame tổng hợp, làm sạch dữ liệu và lưu vào tệp result.csv.
+ Đọc dữ liệu từ table1.csv làm gốc.
+ Lặp qua các tệp CSV từ table3.csv đến table10.csv và nối thêm vào result.
+ Kết hợp với table2.csv để lấy dữ liệu về thủ môn, đảm bảo chỉ giữ các cột cần thiết.
+ Loại bỏ các cột thừa như Unnamed: 0, minutes_90s, birth_year, matches.
+ Xử lý cột minutes để loại bỏ dấu phẩy và chuyển sang kiểu số nguyên.
+ Lọc dữ liệu giữ lại những cầu thủ có số phút thi đấu > 90.
+ Sắp xếp dữ liệu theo player và age.

4. Hàm __main

+ Tải về và xử lý dữ liệu từ URL chính, tạo table1.csv ban đầu.
+ Gọi hàm crawler cho từng URL và ID của bảng tương ứng từ danh sách urls và ids để tạo các bảng bổ sung.
+ Gọi hàm clean_data để hợp nhất các bảng đã tạo và xử lý.

Câu 2:

2.1:

+ Mục đích chính: Đoạn code này được sử dụng để phân tích dữ liệu từ một file CSV chứa thông tin về cầu thủ bóng đá. Nó tìm ra top 3 cầu thủ có điểm số cao nhất và thấp nhất cho từng chỉ số trong một danh sách các chỉ số được xác định trước.

Các bước thực hiện:

+ Đọc dữ liệu từ file CSV: pandas được sử dụng để đọc dữ liệu từ tệp result.csv vào một DataFrame (df), giúp xử lý dữ liệu dễ dàng.
+ In danh sách các cột: Đoạn mã in ra danh sách các cột hiện có trong DataFrame để kiểm tra cấu trúc dữ liệu.
+ Hàm get_top_and_bottom:
  Hàm này nhận vào DataFrame và tên một chỉ số cụ thể.
  Chuyển đổi các giá trị không thể chuyển thành số trong cột chỉ số thành NaN để tránh lỗi tính toán.
  Loại bỏ các hàng có giá trị NaN trong cột được phân tích để đảm bảo dữ liệu chính xác.
  Tìm và trả về top 3 cầu thủ có điểm cao nhất và thấp nhất dựa trên chỉ số đó.
+ Danh sách chỉ số:
  Một danh sách lớn các chỉ số liên quan đến hiệu suất cầu thủ được định nghĩa trước để phân tích, như số bàn thắng, kiến tạo, các chỉ số phòng ngự, v.v.
+ Vòng lặp qua các chỉ số:
  Với mỗi chỉ số trong danh sách, nếu cột đó tồn tại trong DataFrame, đoạn code sẽ gọi hàm get_top_and_bottom và lưu kết quả vào một từ điển.
  Nếu cột không tồn tại, thông báo sẽ được in ra để chỉ rõ cột đó không có trong dữ liệu.
+ In kết quả:
  Kết quả phân tích bao gồm việc in danh sách top 3 cầu thủ có điểm cao nhất và thấp nhất cho từng chỉ số, bao gồm thông tin như tên cầu thủ, đội bóng, quốc tịch, vị trí thi đấu, độ tuổi, và giá trị chỉ số.

2.2:

+ Mục đích chính: Đoạn code này được thiết kế để đọc dữ liệu từ một file CSV, tính toán các thống kê mô tả cho các chỉ số thể thao của cầu thủ, và ghi các kết quả này vào một file CSV mới. Nó cung cấp cả thống kê tổng thể và thống kê theo từng đội.

Chi tiết các bước thực hiện:

+ Đọc dữ liệu từ file CSV: Sử dụng thư viện pandas, code đọc dữ liệu từ file CSV vào một DataFrame tên là df. Đây là bước đầu tiên để thao tác với dữ liệu.
+ In danh sách các cột: Mục đích của việc in ra danh sách các cột là để kiểm tra xem file CSV có cấu trúc dữ liệu như mong đợi hay không.
+ Kiểm tra cột 'team': Đoạn code đảm bảo rằng cột team tồn tại trong DataFrame. Nếu không, một lỗi sẽ được ném ra để cảnh báo người dùng rằng cột này cần thiết cho việc phân tích.
+ Lọc các cột số: Sử dụng phương thức select_dtypes, đoạn code tìm các cột có kiểu dữ liệu số (numeric) và lưu chúng vào danh sách numeric_columns_list. Điều này giúp xác định các chỉ số mà ta có thể thực hiện các phép tính thống kê.
+ Tính toán thống kê tổng thể:

  Tính toán trung vị (median), trung bình (mean), và độ lệch chuẩn (std) cho tất cả các chỉ số của cầu thủ. Các kết quả này được làm tròn đến 2 chữ số thập phân.

  Các giá trị này sau đó được lưu vào một DataFrame mới (overall_df), cùng với số thứ tự (STT) và giá trị team là 'all'.
+ Tính toán thống kê theo đội:

  Sử dụng phương thức groupby, đoạn code tính toán trung vị, trung bình, và độ lệch chuẩn cho mỗi chỉ số theo từng đội. Kết quả này được lưu vào các DataFrame riêng biệt và sau đó gộp thành team_df.

  Gộp các bảng kết quả: Hai bảng thống kê tổng thể và theo đội (overall_df và team_df) được gộp lại thành một bảng duy nhất (final_df), sau đó được ghi vào file CSV mới tên là results2.csv.
+ Kiểm tra nội dung file CSV: Cuối cùng, nội dung của file results2.csv được đọc lại và in ra để người dùng có thể xác minh rằng dữ liệu đã được ghi đúng cách.

2.3:

+ Mục đích chính: Đoạn code này được thiết kế để tạo ra các biểu đồ histogram cho các chỉ số thể thao của cầu thủ từ một file CSV, cả cho toàn bộ giải đấu và cho từng đội bóng. Kết quả sẽ được lưu vào một thư mục mới.
+ Chi tiết các bước thực hiện:
+ Nhập các thư viện cần thiết:
  pandas được sử dụng để thao tác với dữ liệu trong file CSV.
  matplotlib.pyplot và seaborn được sử dụng để vẽ các biểu đồ.
  os được sử dụng để làm việc với hệ thống tệp, bao gồm việc tạo thư mục để lưu trữ hình ảnh.
+ Đọc dữ liệu từ file CSV: Dữ liệu từ file result.csv được đọc vào một DataFrame có tên là data.
+ Xác định các cột chỉ số cần phân tích: Danh sách các chỉ số thể thao cần vẽ biểu đồ histogram được lưu trong biến metrics.
+ Tạo thư mục lưu trữ hình ảnh:
  Kiểm tra xem thư mục histograms đã tồn tại chưa. Nếu chưa, nó sẽ được tạo ra để lưu trữ các biểu đồ histogram.
  Vẽ biểu đồ histogram cho từng chỉ số của toàn giải đấu:
  Vòng lặp đầu tiên duyệt qua từng chỉ số trong danh sách metrics.
  Nếu chỉ số tồn tại trong data, một biểu đồ histogram sẽ được tạo ra với hàm sns.histplot của Seaborn, sử dụng giá trị không bị thiếu (dropna()) và vẽ thêm một đường KDE (Kernel Density Estimate) để thể hiện phân phối.
  Biểu đồ sẽ được lưu với tên theo định dạng {metric}_all_players.png.
+ Vẽ biểu đồ histogram cho từng chỉ số của từng đội bóng:
  Sử dụng unique() để lấy danh sách các đội bóng từ cột team.
  Một vòng lặp lồng nhau được sử dụng để tạo biểu đồ cho từng đội. Đầu tiên, đoạn mã lọc ra các cầu thủ thuộc về đội hiện tại (team_subset).
  Tương tự như trước đó, nếu chỉ số tồn tại trong team_subset, biểu đồ histogram cho đội sẽ được tạo và lưu với tên theo định dạng {metric}_{team}.png.
+ Kết thúc: Cuối cùng, một thông báo được in ra cho biết các biểu đồ histogram đã được lưu thành công trong thư mục histograms.

2.4:

+ Mục đích chính: Đoạn code này được thiết kế để tìm đội bóng có điểm số cao nhất cho mỗi chỉ số từ một file CSV. Kết quả sẽ được hiển thị cho mỗi chỉ số mà đội bóng đạt được điểm cao nhất.
+ Chi tiết các bước thực hiện:
+ Nhập thư viện cần thiết:
  pandas được sử dụng để thao tác với dữ liệu từ file CSV.
+ Đọc dữ liệu từ file CSV:
  Dữ liệu từ file result.csv được đọc vào một DataFrame có tên là df.
+ In danh sách các cột:
  In ra danh sách các cột trong DataFrame để kiểm tra.
+ Hàm tìm đội bóng có điểm số cao nhất cho mỗi chỉ số:
  Hàm find_top_team_per_stat(df, column) nhận vào DataFrame và tên cột chỉ số.
  Sử dụng pd.to_numeric để chuyển đổi các giá trị không thể chuyển đổi sang số thành NaN, giúp dễ dàng loại bỏ chúng sau này.
  Dữ liệu sẽ được làm sạch bằng cách loại bỏ các hàng chứa giá trị NaN trong cột được phân tích.
  Nếu không có giá trị hợp lệ nào sau khi làm sạch, hàm sẽ trả về None.
  Tìm đội bóng có chỉ số cao nhất bằng cách sử dụng idxmax() để lấy chỉ số của hàng có giá trị lớn nhất trong cột và sau đó lấy giá trị tương ứng trong cột team.
+ Xác định các chỉ số cần phân tích:
  Một danh sách columns_to_analyze được tạo ra, bao gồm tất cả các chỉ số cần phân tích. Sử dụng set() để loại bỏ các cột trùng lặp.
+ Tạo từ điển để lưu kết quả:
  Một từ điển results sẽ lưu trữ kết quả cho từng chỉ số.
+ Tìm đội bóng có chỉ số cao nhất cho mỗi chỉ số:
  Vòng lặp duyệt qua từng chỉ số trong columns_to_analyze.
  Nếu cột tồn tại trong DataFrame, hàm find_top_team_per_stat sẽ được gọi để tìm đội bóng có chỉ số cao nhất.
  Nếu không có dữ liệu hợp lệ cho một cột, thông báo sẽ được in ra.
+ Hiển thị kết quả:
  Kết quả sẽ được in ra cho từng chỉ số, cho biết đội bóng có điểm số cao nhất và giá trị của chỉ số đó.

Câu 3:

3.1:

+ Mục đích chính: Đoạn code này được thiết kế để áp dụng thuật toán K-means clustering nhằm phân loại cầu thủ dựa trên các chỉ số hiệu suất như số bàn thắng không phạt đền, số bàn thắng phạt đền, số lần kiến tạo và số phút thi đấu.

+ Chi tiết các bước thực hiện:

Nhập các thư viện cần thiết:

pandas để thao tác dữ liệu từ file CSV.

KMeans từ sklearn để thực hiện phân cụm.

matplotlib.pyplot để trực quan hóa dữ liệu.

Bước 1: Đọc dữ liệu từ tệp CSV:

Dữ liệu được đọc từ file result.csv vào một DataFrame có tên là data. In ra danh sách các cột có trong DataFrame để kiểm tra và xác định các chỉ số cần thiết.

Bước 2: Chọn các chỉ số cần thiết để phân loại:

Chọn các cột từ DataFrame mà bạn muốn sử dụng để phân loại, trong trường hợp này là 'non-Penalty Goals', 'Penalty Goals', 'assists', và 'minutes'. Dữ liệu sẽ được lọc để chỉ chứa các hàng không có giá trị NaN bằng cách sử dụng dropna().

Bước 3: Chọn số lượng cụm K:

Số lượng cụm K được thiết lập là 3, có thể điều chỉnh theo nhu cầu và mục tiêu phân tích của bạn.

Bước 4: Khởi tạo mô hình K-means:

Tạo một đối tượng KMeans với số cụm đã chọn và random_state được đặt để đảm bảo tính ngẫu nhiên trong quá trình khởi tạo có thể tái sản xuất.

Bước 5: Huấn luyện mô hình và dự đoán các nhãn:

Sử dụng phương thức fit_predict() để huấn luyện mô hình K-means và dự đoán nhãn cho từng cầu thủ trong features.

Bước 6: Thêm nhãn cụm vào DataFrame gốc:

Nhãn cụm dự đoán được thêm vào DataFrame gốc, tạo một cột mới có tên là 'Cluster'.

Bước 7: Hiển thị kết quả:

In ra danh sách cầu thủ cùng với nhãn cụm của họ để dễ dàng theo dõi.

Bước 8: Trực quan hóa:

Sử dụng matplotlib để vẽ biểu đồ phân tán (scatter plot) với các cầu thủ được phân loại dựa trên số bàn thắng không phạt đền và số lần kiến tạo. Mỗi màu trong biểu đồ tương ứng với một cụm khác nhau, và thêm nhãn cho các trục cùng với tiêu đề và thanh màu (colorbar).

3.2:

+ Bước 1: Đọc Dữ Liệu

Sử dụng thư viện pandas để đọc dữ liệu từ tệp CSV. Điều này cho phép dễ dàng quản lý và xử lý dữ liệu trong Python.

+ Bước 2: Chọn Các Chỉ Số Cần Thiết

Chọn các chỉ số cụ thể từ dữ liệu để phân loại. Các chỉ số này bao gồm:

Số bàn thắng không phải penalty.
Số bàn thắng penalty.
Số lần hỗ trợ.
Số phút thi đấu.

Việc lựa chọn này rất quan trọng để đảm bảo mô hình học được những đặc điểm chính của dữ liệu.

+ Bước 3: Tính WCSS cho Các Giá Trị K Khác Nhau

Tạo một danh sách để lưu trữ giá trị WCSS (Within-Cluster Sum of Squares) cho các giá trị K từ 1 đến 10.

Thực hiện phân cụm K-means cho từng giá trị K và lưu trữ WCSS tương ứng.

WCSS được sử dụng để đánh giá mức độ phân tán của các điểm trong cùng một cụm; giá trị càng nhỏ thể hiện rằng các điểm trong cùng một cụm càng gần nhau.

+ Bước 4: Vẽ Biểu Đồ Elbow

Sử dụng matplotlib để vẽ biểu đồ Elbow, giúp xác định số lượng cụm K tối ưu.

Biểu đồ này thể hiện mối quan hệ giữa số lượng cụm K và giá trị WCSS, giúp dễ dàng xác định điểm "góc" nơi WCSS bắt đầu giảm chậm lại.

+ Bước 5: Chọn Số Lượng Cụm K Tối Ưu

Dựa trên biểu đồ Elbow, chọn giá trị K tối ưu cho mô hình. Ví dụ, K có thể được chọn là 3 nếu đó là điểm góc trên biểu đồ.

+ Bước 6: Khởi Tạo Mô Hình K-means

Khởi tạo mô hình K-means với số cụm K đã chọn. Điều này chuẩn bị cho việc huấn luyện mô hình.

+ Bước 7: Huấn Luyện Mô Hình và Dự Đoán Các Nhãn

Huấn luyện mô hình K-means với các chỉ số đã chọn và dự đoán nhãn cho từng điểm dữ liệu.

Kết quả là mỗi cầu thủ sẽ được gán một nhãn cụm, phản ánh đặc điểm của họ trong không gian phân loại.

+ Bước 8: Thêm Nhãn Cụm vào DataFrame Gốc

Thêm nhãn cụm đã dự đoán vào DataFrame gốc để có thể dễ dàng theo dõi và phân tích kết quả phân cụm.

+ Bước 9: Hiển Thị Kết Quả

In ra kết quả phân loại các cầu thủ theo cụm. Điều này cho phép người dùng thấy được cách mà các cầu thủ được nhóm lại với nhau.

3.3:

Mô tả từng bước chi tiết:

+ Tải dữ liệu: Sử dụng pd.read_csv để tải dữ liệu từ tệp result.csv.
+ Lựa chọn các cột: Chọn các cột phù hợp từ tập dữ liệu để phân tích ('non-Penalty Goals', 'Penalty Goals', 'assists', 'minutes').
+ Chuẩn hóa dữ liệu: Áp dụng StandardScaler để chuẩn hóa dữ liệu, đảm bảo các cột có đơn vị và khoảng giá trị tương đồng.
+ Áp dụng PCA: Sử dụng PCA để giảm chiều dữ liệu xuống còn 2 thành phần chính (PC1 và PC2), giúp trực quan hóa dữ liệu dễ dàng hơn.
+ Phân cụm: Dùng KMeans để phân cụm dữ liệu thành 3 nhóm dựa trên các đặc trưng đã chuẩn hóa.
+ Vẽ biểu đồ: Sử dụng matplotlib để vẽ biểu đồ phân tán, hiển thị các nhóm cầu thủ trên mặt phẳng 2D với các màu khác nhau.
+ Biểu đồ này giúp phân tích trực quan cách các cầu thủ được phân nhóm theo các đặc trưng chính, hỗ trợ việc phân tích dữ liệu và xác định mẫu trong tập dữ liệu.

3.4:

+ Thư viện:
  pandas, numpy để xử lý dữ liệu.
  matplotlib để vẽ biểu đồ.
  argparse để xử lý đầu vào từ dòng lệnh (trong trường hợp cần mở rộng).
+ Hàm radar_chart:
+ Đọc dữ liệu: Dùng pd.read_csv để đọc dữ liệu từ file result.csv.
+ Trích xuất dữ liệu cầu thủ: Dùng data[data['player'] == player_name] để lấy chỉ số cụ thể của từng cầu thủ.
+ Kiểm tra tồn tại: Đảm bảo rằng dữ liệu của cả hai cầu thủ tồn tại trong tập dữ liệu.
+ Xử lý dữ liệu: Sắp xếp góc và đảm bảo dữ liệu đóng vòng để vẽ biểu đồ radar.
+ Vẽ biểu đồ:
    Sử dụng plt.subplots với polar=True để tạo biểu đồ radar.
    Dùng ax.fill để điền các vùng dữ liệu, với các màu khác nhau cho từng cầu thủ.
    Cài đặt nhãn và chú thích: Đặt tiêu đề, các nhãn chỉ số, và chú thích để dễ hiểu hơn.
+ Phần __main:
  Nhận đầu vào từ người dùng thông qua input() để chọn cầu thủ và các chỉ số cần so sánh.
  Chuyển đổi chuỗi nhập vào thành một danh sách các chỉ số.
  Gọi hàm radar_chart để vẽ biểu đồ.