HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH VỚI PYTHON

BÁO CÁO BÀI TẬP LỚN

LẬP TRÌNH VỚI PYTHON

Tên sinh viên: Phan Văn Bang

Mã sinh viên: B22DCCN057

Nhóm lớp học: *Nhóm 11*

Số điện thoại: 0332936637

HÀ NỘI, THÁNG 11/2024

Lời nói đầu

Bài tập lớn này được thực hiện nhằm mục đích củng cố và nâng cao kiến thức về lập trình Python, cũng như rèn luyện kỹ năng áp dụng ngôn ngữ vào giải quyết các bài toán thực tiễn. Trong quá trình thực hiện, em đã tiếp cận với nhiều khía cạnh quan trọng của Python, từ cấu trúc dữ liệu cơ bản, hàm và lớp đối tượng, đến các thư viện nâng cao phục vụ cho việc xử lý dữ liệu, thao tác với file, và tạo giao diện người dùng.

Mục tiêu chính của bài tập là xây dựng một ứng dụng quản lý và phân tích dữ liệu với các yêu cầu cụ thể, giúp người dùng thu thập, xử lý và xuất dữ liệu theo cách tối ưu. Qua đó, em đã có cơ hội tìm hiểu và sử dụng các thư viện quan trọng như pandas, openpyxl, và csv để xử lý dữ liệu và lưu trữ, cũng như hiểu rõ hơn về quy trình xây dựng một phần mềm hoàn chỉnh từ thiết kế đến triển khai.

Trong quá trình thực hiện bài tập lớn này, em đã gặp không ít khó khăn và thử thách, từ việc tối ưu mã, xử lý các lỗi thường gặp, đến cách tổ chức và quản lý dữ liệu hiệu quả. Tuy nhiên, nhờ vào sự hướng dẫn của giảng viên và tài liệu tham khảo phong phú, em đã vượt qua được các khó khăn và hoàn thành bài tập với những kiến thức bổ ích.

Em hy vọng rằng bài báo cáo này sẽ giúp người đọc có cái nhìn rõ nét về quá trình xây dựng ứng dụng bằng Python, đồng thời cũng cung cấp một số thông tin hữu ích cho các bạn sinh viên, những người mới bắt đầu với ngôn ngữ này.

Phần I: Viết chương trình Python thu thập dữ liệu phân tích cầu thủ

1. Các thư viện cần import để lấy và xử lí dữ liệu: requests, BeautifulSoup, Selenium, pandas, numpy, matplotlib.pyplot, seaborn, lxml, re, time, json, …
2. Yêu cầu về thu thập dữ liệu:
   - Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.
   - Nguồn dữ liệu: https://fbref.com/en/
   - Ghi kết quả ra file 'results.csv', bảng kết quả có cấu trúc như sau:
     + Mỗi cột tương ứng với một chỉ số.
     + Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.
     + Chỉ số nào không có hoặc không áp dụng thì để là N/a
   - Các chỉ số cần thống kê: 171 chỉ số.
3. Các bước thực hiện lấy dữ liệu từ trang web về:

Bước 1: Nhập các định nghĩa từ các tệp python khác:

+ Player và Player_Manager: Dùng để tạo các đối tượng cầu thủ và quản lý danh sách các cầu thủ.

+ header và row từ tieu_de: Để tạo tiêu đề và định dạng dữ liệu của các hàng trong tệp CSV.

⇨ Ưu điểm: code được format dễ xem và sửa chữa nếu xảy ra lỗi.

⇨ Nhược điểm: Tốn thời gian và tài nguyên.

Bước 2: Định nghĩa hàm GetDataFromWeb:

+ Hàm này nhận các tham số:

- url: URL của trang cần lấy dữ liệu.
- Xpath_player: Xpath của bảng dữ liệu cần lấy.
- Data_Name: Tên dữ liệu (ví dụ: “Standard”, “Goalkeeping”).

+ Mở trình duyệt Chrome, truy cập URL, sau đó lấy bảng dữ liệu theo Xpath đã chỉ định.

❖ **Lưu ý**: Có thời gian chờ time.sleep(5) để đảm bảo trang tải xong.

+ Duyệt qua từng hàng và cột trong bảng:

- Lấy thông tin về tên, đội, và các chỉ số của cầu thủ.
- Chuyển đổi chuỗi số (có dấu phẩy) thành số thực để dễ tính toán.

+ Đóng trình duyệt và trả về dữ liệu đã thu thập.

Bước 3: Lấy dữ liệu từ các trang và lưu vào Player_Manager:

+ Gọi hàm GetDataFromWeb với các URL khác nhau để lấy các loại chỉ số khác nhau (vd: “Goalkeeping”, “Shooting”, “Passing”).

+ Đối với mỗi cầu thủ, hàm findPlayerByNameandTeam sẽ kiểm tra xem cầu thủ đã tồn tại trong player_manager hay chưa:

- Nếu chưa có, tạo đối tượng cầu thủ mới và thêm vào player_manager.
- Nếu đã có, cập nhật các chỉ số (sút, chuyền bóng, v.v.) vào đối tượng cầu thủ đó.

+ Sau khi lấy đủ dữ liệu, dùng player_manager.sortingByName() để sắp xếp danh sách cầu thủ theo tên.

Bước 4: Lưu dữ liệu vào file csv:

+ Mở tệp CSV ở chế độ ghi (w), viết tiêu đề và lần lượt ghi từng hàng dữ liệu:

- Sử dụng hàm header để tạo dòng tiêu đề.
- Với mỗi cầu thủ trong player_manager.list_player, hàm row được dùng để lấy thông tin đã định dạng và ghi vào tệp CSV.

+ Cuối cùng, thông báo "Exam 1 Success - Player Data Saved as CSV" để xác nhận quá trình thành công.

4. Kết quả cần đạt được:

- Lấy được 1 bảng dữ liệu có đủ 171 cols và 494 rows (tính cả tiêu đề).
- Tên các cột phải có đánh dấu rõ rang thuộc bảng nào để tránh trùng lặp dữ liệu.

Phần II: Xử lí dữ liệu

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

Bước 1: Đọc file CSV và chuẩn bị dữ liệu.

Bước 2: Kiểm tra sự tồn tại của cột 'name':

- Đoạn mã kiểm tra xem cột 'name' có trong dữ liệu hay không. Nếu không, chương trình sẽ in ra thông báo và dừng lại.

Bước 3: Định nghĩa hàm collect_top_bottom:

- Hàm này được dùng để thu thập top và bottom n (ở đây là 3) giá trị trong một cột số cụ thể.
- Lấy dữ liệu của cột 'name' và cột số đang xét để đảm bảo tính toán chính xác.
- Kiểm tra xem cột có đủ dữ liệu để lấy top/bottom không. Nếu không đủ, in thông báo lỗi.

- Sử dụng nlargest và nsmallest để lấy các giá trị lớn nhất và nhỏ nhất trong cột.
- Thêm dữ liệu top/bottom vào danh sách results dưới dạng từ điển với các thông tin về loại (Type), tên (name), điểm số (Score), và tên cột (Metric).
- Thêm dòng trống để phân tách các nhóm dữ liệu (top và bottom) trong bảng kết quả.

Bước 4: Duyệt qua các cột dạng số từ cột thứ 5 trở đi.

Bước 5: Lưu vào file csv.

2.Tìm trung vị của mỗi chỉ số:

Bước 1: Đọc file CSV:

- Đọc dữ liệu từ file result.csv tại đường dẫn đã chỉ định, với giả định file có tiêu đề ở hàng đầu tiên.

Bước 2: Tạo DataFrame để lưu kết quả cuối cùng:

- Khởi tạo danh sách final_results để lưu các kết quả sẽ được tính toán sau đó.

Bước 3: Duyệt qua từng cột thuộc tính (bắt đầu từ cột thứ 4):

- Bắt đầu từ cột thứ 4 của bảng (giả định các cột từ 1-3 là các cột thông tin khác).
- Lấy từng cột và chuyển đổi dữ liệu về dạng số bằng pd.to_numeric để tránh lỗi khi tính toán.

Bước 4: Tính toán median, mean, và std cho toàn bộ dữ liệu trong cột:

- Tính trung vị (median), trung bình (mean), và độ lệch chuẩn (std) cho toàn bộ giá trị trong cột.
- Nếu cột đang xử lý là cột đầu tiên (cột thứ 4 trong bảng), thêm các giá trị tính toán vào final_results dưới dạng một hàng với tiêu đề 'All'.
- Nếu không, mở rộng hàng đầu tiên của final_results với các giá trị tính toán mới.

Bước 5: Tính toán cho từng đội:

- Sử dụng groupby theo tên đội (team) để chia dữ liệu thành từng nhóm.
- Với mỗi đội, tính các giá trị trung vị, trung bình và độ lệch chuẩn cho cột hiện tại.
- Kiểm tra nếu đội đã có trong final_results:
  - Nếu đã có, mở rộng hàng tương ứng của đội đó với các chỉ số mới.
  - Nếu chưa có, tạo một hàng mới cho đội đó và thêm vào final_results.

Bước 6: Chuyển danh sách kết quả cuối cùng thành DataFrame:

- Sau khi hoàn thành việc tính toán cho tất cả các cột và các đội, chuyển final_results thành DataFrame final_df.

Bước 7: Cập nhật tiêu đề cột:

- Tạo danh sách column_titles bắt đầu với 'team'.
- Thêm tiêu đề mới cho từng cột thuộc tính, mỗi thuộc tính có 3 tiêu đề con (Median, Mean, Std) ứng với từng giá trị tính toán.

Bước 8: Gán tiêu đề cột cho DataFrame kết quả:

- Gán column_titles làm tiêu đề cột cho final_df.

Bước 9: Lưu kết quả ra file CSV:

- In final_df ra màn hình để kiểm tra kết quả.
- Lưu final_df vào file results2.csv tại đường dẫn đã chỉ định.

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

Bước 1: Đọc và xử lý dữ liệu từ file CSV

- Đọc tệp CSV: Đọc dữ liệu từ file result.csv và lưu vào DataFrame df.
- Làm sạch tên cột: Loại bỏ khoảng trắng và các phần không cần thiết như "Unnamed:" khỏi tên cột.
- Kiểm tra tên cột: In ra tên các cột để kiểm tra.

Bước 2: Tính trung vị, trung bình và độ lệch chuẩn cho mỗi chỉ số

- Khởi tạo một DataFrame rỗng final_results để lưu các kết quả tính toán.
- Duyệt qua từng cột thuộc tính:
  - Với mỗi cột (bắt đầu từ cột thứ 4), chuyển đổi dữ liệu về dạng số để có thể tính toán.
  - Tính trung vị (median), trung bình (mean), và độ lệch chuẩn (std) cho toàn bộ dữ liệu trong cột đó.
  - Lưu kết quả tính toán của toàn giải cho cột đầu tiên vào final_results với tiêu đề "All". Đối với các cột khác, mở rộng hàng đầu tiên bằng các giá trị tính toán mới.
- Tính toán cho từng đội:
  - Sử dụng groupby theo tên đội (team) để chia dữ liệu thành từng nhóm.
  - Với mỗi đội, tính trung vị, trung bình và độ lệch chuẩn của cột hiện tại.
  - Kiểm tra xem đội đã có trong final_results chưa:
    - Nếu đã có, mở rộng hàng của đội với các chỉ số mới.
    - Nếu chưa có, thêm một hàng mới cho đội đó trong final_results.

Bước 3: Chuyển danh sách kết quả thành DataFrame và lưu vào CSV

- Chuyển final_results thành DataFrame final_df.
- Cập nhật tiêu đề cột: Tạo danh sách column_titles với tên các chỉ số để đảm bảo mỗi cột có tiêu đề phù hợp.
- Gán tiêu đề cột: Gán column_titles làm tiêu đề cột cho final_df.
- Lưu kết quả ra file CSV: In final_df để kiểm tra và lưu vào results2.csv.

Bước 4: Vẽ histogram phân bố của mỗi chỉ số

- Lọc các cột dạng số: Xác định các cột có kiểu số (float và int) để vẽ biểu đồ.
- Vẽ histogram cho từng thuộc tính của toàn giải:
  - Với mỗi thuộc tính số, tạo biểu đồ histogram sử dụng sns.histplot, thêm đường kde để hiển thị phân phối.
  - Đặt tiêu đề, nhãn trục x và y cho biểu đồ, sau đó hiển thị.
- Vẽ histogram cho từng đội:
  - Sử dụng sns.FacetGrid để tạo biểu đồ histogram cho từng đội, sắp xếp theo từng cột và hiển thị trong cùng một biểu đồ lớn.
  - Đặt nhãn và tiêu đề cho mỗi biểu đồ nhỏ tương ứng với từng đội.

4.Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024 ?

- Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số:

Bước 1: Đọc dữ liệu từ file CSV:

- Đọc dữ liệu từ file CSV chứa thông tin về các đội bóng và các chỉ số khác nhau.

Bước 2: Lọc bỏ hàng "All":

- Nếu file dữ liệu có chứa hàng "All" (đại diện cho tổng thể hoặc trung bình của tất cả các đội), loại bỏ hàng này để tránh ảnh hưởng đến kết quả tìm kiếm đội có điểm số cao nhất.

Bước 3: Tạo dictionary để lưu kết quả:

- Khởi tạo một dictionary (max_scores) để lưu tên đội bóng và giá trị điểm số cao nhất cho mỗi chỉ số.

Bước 4: Duyệt qua các cột chỉ số:

- Bỏ qua cột team vì đây là cột chứa tên đội bóng.
- Đối với mỗi cột chỉ số khác (các cột từ vị trí thứ hai trở đi), tìm giá trị lớn nhất và đội bóng đạt giá trị đó.

Bước 5: Lấy đội bóng và giá trị điểm cao nhất:

- Sử dụng phương thức idxmax() để lấy chỉ số hàng của đội có giá trị cao nhất cho chỉ số đang xét.
- Lưu tên đội bóng và giá trị cao nhất của chỉ số này vào dictionary max_scores.

Bước 6: In kết quả:

- Duyệt qua max_scores và in ra đội bóng có điểm số cao nhất cho mỗi chỉ số, kèm giá trị cụ thể.

- Theo em thì Manchester City là đội bóng có phong độ tốt nhất giải ngoại hạng Anh mùa 2023-2024. Vì có Đội hình chất lượng và chiều sâu đội ngũ, Kỷ luật chiến thuật và sự nhất quán, Cơ sở hạ tầng và đầu tư, Phân tích dữ liệu và chuẩn bị kỹ lưỡng.

Phần III: Xử lí dữ liệu

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

Bước 1: Import thư viện cần thiết

- Nhập các thư viện cần thiết, bao gồm pandas, numpy, seaborn, matplotlib.pyplot để xử lý dữ liệu và vẽ đồ thị.
- Tắt các cảnh báo không cần thiết bằng warnings.filterwarnings('ignore').

Bước 2: Đọc dữ liệu từ file CSV

- Đọc dữ liệu từ result.csv và lưu vào DataFrame df.
- Hiển thị thông tin dữ liệu:
  - df.head(10): Xem 10 dòng đầu của dữ liệu.
  - df.shape: Xem kích thước của dữ liệu.
  - df.info(): Kiểm tra tổng quan về các cột, kiểu dữ liệu, và số lượng giá trị null.
  - df.isna().sum(): Đếm số giá trị thiếu (null) trong mỗi cột.
  - df['team'].value_counts(): Xem số lượng các đội trong dữ liệu.
  - df.describe().T: Thống kê tổng quan cho các cột số (trung bình, độ lệch chuẩn, min, max, v.v.).

Bước 3: Chuẩn bị dữ liệu cho K-means Clustering

- Xác định tập dữ liệu đầu vào: Chọn các cột có kiểu số (float và int) để làm dữ liệu cho mô hình K-means và lưu vào biến X.

Bước 4: Chạy mô hình K-means với các giá trị k từ 2 đến 9

- Khởi tạo các danh sách trống sse và silhouette_scores để lưu giá trị SSE và Silhouette Score cho mỗi số cụm.
- Lặp qua từng giá trị k trong khoảng từ 2 đến 9:
  - Khởi tạo mô hình KMeans với số cụm k và random_state=42 để tái tạo được kết quả.
  - Huấn luyện mô hình với dữ liệu X bằng cách gọi kmeans.fit(X).
  - Tính toán SSE (Sum of Squared Errors): Lấy độ biến thiên trong mỗi cụm và lưu vào danh sách sse.
  - Tính toán Silhouette Score: Đo lường mức độ phân tách giữa các cụm và lưu vào silhouette_scores.

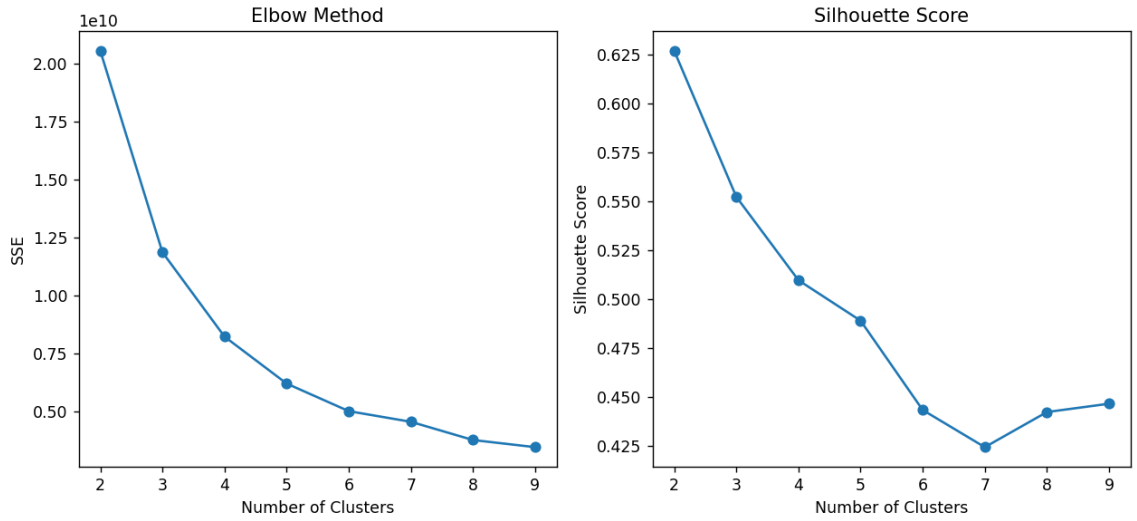
Bước 5: Vẽ đồ thị Elbow và Silhouette Score

- Elbow Method: Vẽ đồ thị biểu diễn mối quan hệ giữa số cụm và SSE. Đồ thị "Elbow" giúp xác định số cụm tối ưu bằng cách tìm điểm gãy khúc trong đường cong SSE.
  - Trục x: Số cụm (k).
  - Trục y: SSE (Sum of Squared Errors).
  - Tiêu đề: "Elbow Method".
- Silhouette Score: Vẽ đồ thị biểu diễn mối quan hệ giữa số cụm và Silhouette Score. Giá trị Silhouette càng cao cho thấy sự phân cụm tốt hơn.
  - Trục x: Số cụm (k).
  - Trục y: Silhouette Score.
  - Tiêu đề: "Silhouette Score".
- Hiển thị cả hai biểu đồ trên cùng một hàng với plt.subplot.

Bước 6: Hiển thị đồ thị

- Sử dụng plt.show() để hiển thị đồ thị Elbow và Silhouette Score, giúp lựa chọn số cụm tối ưu cho mô hình K-means.

⇨ Kết quả:



2. Nhìn vào đồ thị chọn cluster = 4:

Quan điểm 1: Phương pháp Elbow (Khuỷu tay)

- Ý tưởng: Đồ thị Elbow giúp xác định số cụm tối ưu bằng cách quan sát độ biến thiên của SSE (Tổng bình phương sai số) khi tăng số lượng cụm kkk.
- Quan sát: Khi vẽ đồ thị Elbow, chúng ta thấy rằng SSE giảm mạnh khi số cụm tăng lên, nhưng sau một mức nào đó (thường gọi là "khuỷu tay"), mức giảm chậm lại. Điểm khuỷu tay này gợi ý rằng thêm cụm không cải thiện đáng kể việc phân cụm.
- Áp dụng trong bài toán này: Đồ thị cho thấy điểm khuỷu tay có thể nằm ở k=4. Với $k \geq 4$, việc thêm cụm làm giảm SSE rất ít, có nghĩa là cụm thứ 5, 6, v.v., sẽ không giúp mô hình tốt hơn nhiều. Điều này chỉ ra rằng k=4 là số cụm hợp lý để phân chia dữ liệu.

Quan điểm 2: Silhouette Score (Điểm Silhouette)

- Ý tưởng: Silhouette Score đo lường chất lượng phân cụm bằng cách đánh giá khoảng cách giữa các điểm trong cùng cụm và giữa các cụm khác nhau. Giá trị Silhouette cao thể hiện rằng các điểm trong cụm gần nhau và các cụm cách biệt rõ ràng.
- Quan sát: Khi kiểm tra đồ thị Silhouette Score, bạn có thể thấy rằng tại k=4, giá trị Silhouette Score đạt mức khá cao, cho thấy rằng các cụm có sự phân biệt tốt với nhau. Nếu chọn k<4, các cụm có thể chồng lấn nhau và không phân biệt rõ ràng, dẫn đến Silhouette Score thấp hơn.

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

- PCA được import từ sklearn.decomposition và seaborn để vẽ biểu đồ trực quan.

- Giảm số chiều với PCA:

```
# Giảm số chiều với PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

- n_components=2 chỉ định rằng chúng ta muốn giảm dữ liệu về 2 chiều (Component 1 và Component 2).
- X_pca là dữ liệu sau khi đã giảm chiều từ dữ liệu gốc X.
- fit_transform là phương thức vừa "học" các thành phần chính từ dữ liệu X, vừa áp dụng để chuyển đổi dữ liệu sang không gian mới với 2 chiều.

- Áp dụng K-means trên dữ liệu PCA:

```
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(X_pca)
```

- n_clusters=4 chỉ định rằng bạn muốn phân cụm dữ liệu thành 4 cụm.
- fit_predict là phương thức vừa huấn luyện mô hình K-means vừa trả về nhãn cụm cho mỗi điểm dữ liệu (dưới dạng chỉ số từ 0 đến 3 cho 4 cụm).
- clusters là mảng chứa nhãn cụm cho mỗi điểm, thể hiện điểm đó thuộc về cụm nào.

- Vẽ biểu đồ phân cụm:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=clusters,
palette="viridis", legend='full')
```

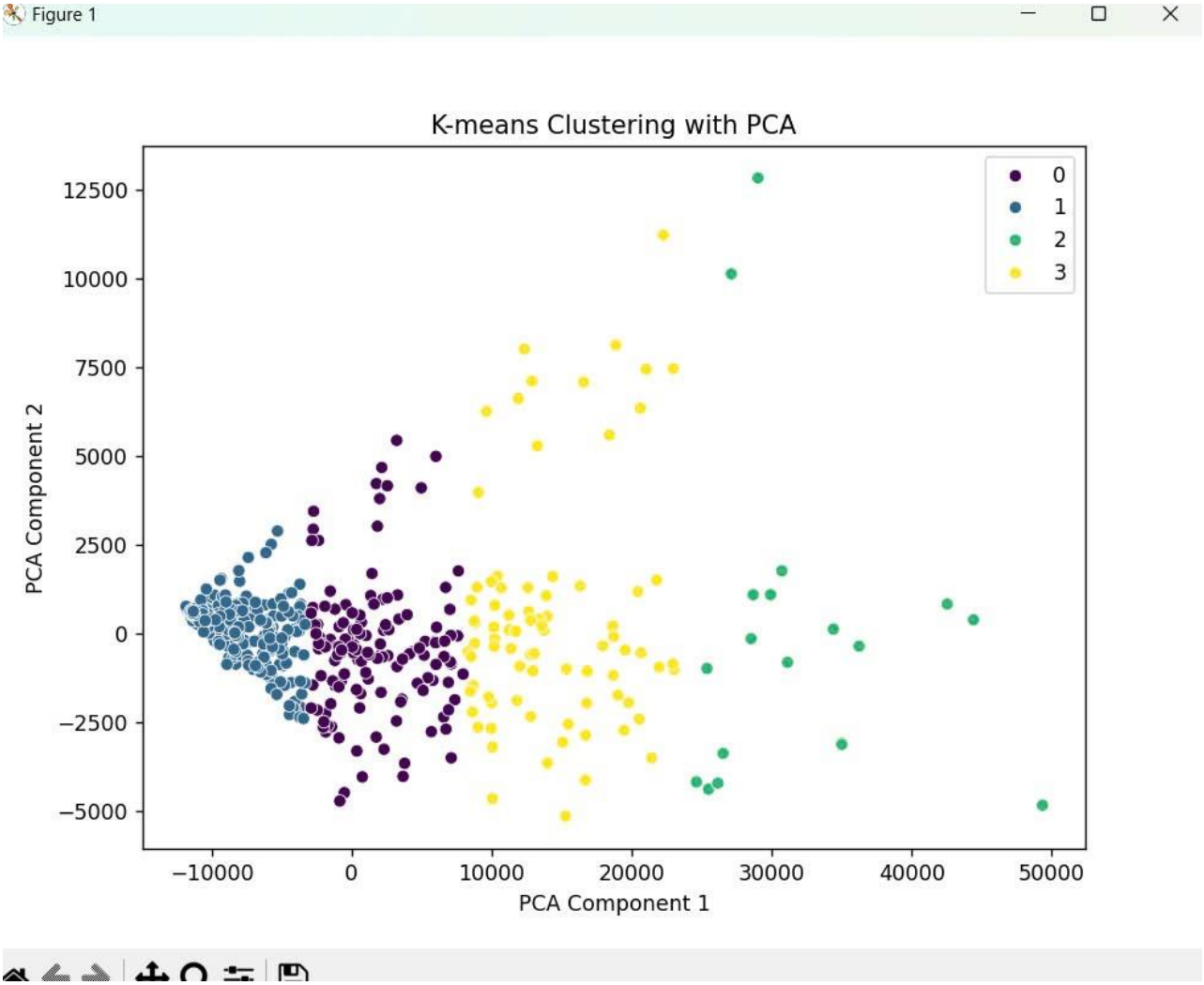
- sns.scatterplot vẽ biểu đồ tán sắc (scatter plot) cho dữ liệu đã được giảm chiều.
- x=X_pca[:, 0] và y=X_pca[:, 1] biểu thị hai trục của PCA (Component 1 và Component 2).
- hue=clusters giúp phân biệt các cụm bằng màu sắc khác nhau.
- palette="viridis" đặt bảng màu "viridis" để biểu diễn các cụm khác nhau.
- legend='full' hiển thị đầy đủ chú thích cho các cụm.

- Tùy chỉnh và hiển thị biểu đồ:

```
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("K-means Clustering with PCA")
plt.show()
```

- Đặt nhãn cho trục x và trục y, cũng như tiêu đề cho biểu đồ.
- plt.show() hiển thị biểu đồ.

- Kết quả:



4. Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ với đầu vào:

- Nhập thư viện và đọc dữ liệu:
  - matplotlib.pyplot để vẽ biểu đồ.
  - pandas và numpy để xử lý dữ liệu.
  - df = pd.read_csv(...): Đọc dữ liệu từ tệp CSV và lưu vào DataFrame df.
- Thiết lập tên cầu thủ và các chỉ số cần so sánh:

```
p1 = "Max Aarons"  # Thay tên cầu thủ
p2 = "Tyler Adams"  # Thay tên cầu thủ
attributes = ["npxG", "Long_Cmp", "Pass_Cmp_outcome"]  # Thay đổi danh sách các chỉ số muốn so sánh
```

  - p1 và p2 là tên hai cầu thủ cần so sánh.
  - attributes là danh sách các chỉ số muốn so sánh giữa hai cầu thủ này.
- Trích xuất dữ liệu của các cầu thủ:

```
# Trích xuất dữ liệu
data1 = df[df['name'] == p1][attributes].values.flatten()
data2 = df[df['name'] == p2][attributes].values.flatten()
```

  - Lọc dữ liệu trong DataFrame df để lấy các chỉ số cho cầu thủ p1 và p2.

- values.flatten() giúp chuyển đổi dữ liệu thành một mảng phẳng để dễ dàng sử dụng cho vẽ biểu đồ.

- Thiết lập biểu đồ radar:

```
86    # Thiết lập biểu đồ radar
87    labels = np.array(attributes)
88    num_vars = len(attributes)
```

- labels là danh sách các chỉ số sẽ làm nhãn trên biểu đồ radar.
- num_vars là số lượng chỉ số, sẽ là số cạnh trên biểu đồ radar.

- Tạo các góc cho biểu đồ radar:

```
# Tạo các góc cho biểu đồ radar
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
data1 = np.concatenate((data1, [data1[0]]))  # Vòng lại điểm đầu
data2 = np.concatenate((data2, [data2[0]]))  # Vòng lại điểm đầu
angles += angles[:1]
```

- angles là một mảng các góc cho mỗi chỉ số trên biểu đồ radar.
- np.linspace giúp tạo một mảng các góc đều nhau từ 0 đến 2π2\pi2π.
- np.concatenate((data1, [data1[0]])) nối giá trị đầu tiên vào cuối mảng data1 và data2 để biểu đồ radar được vẽ trọn vẹn thành hình đa giác.
- angles += angles[:1] để đóng vòng góc lại, đảm bảo biểu đồ kết nối điểm cuối với điểm đầu.

- Vẽ biểu đồ radar:

```
96    # Vẽ biểu đồ radar
97    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
98    ax.fill(angles, data1, color='blue', alpha=0.25, label=p1)
99    ax.fill(angles, data2, color='red', alpha=0.25, label=p2)
00    ax.plot(angles, data1, color='blue', linewidth=2)
01    ax.plot(angles, data2, color='red', linewidth=2)
```

- subplot_kw=dict(polar=True) để tạo biểu đồ radar (biểu đồ cực).
- ax.fill để vẽ các vùng màu cho các chỉ số của từng cầu thủ, giúp trực quan hóa vùng bao phủ của từng cầu thủ trên biểu đồ.
- ax.plot vẽ đường nối các điểm trên biểu đồ cho từng cầu thủ.

- Thiết lập các nhãn:
  - ax.set_yticklabels([]) để ẩn nhãn của trục y (thang đo bán kính).
  - ax.set_xticks và ax.set_xticklabels để đặt các nhãn của chỉ số tại các góc trên biểu đồ radar.
- Thêm tiêu đề và chú thích:
  - Đặt tiêu đề cho biểu đồ để biết rằng đây là so sánh giữa hai cầu thủ p1 và p2.

- Thêm chú thích để biết mỗi màu đại diện cho cầu thủ nào.

⇨ Kết quả: