BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

------📖-----

BÁO CÁO BÀI TẬP LỚN

MÔN: LẬP TRÌNH PYTHON

Giảng viên hướng dẫn:	Kim Ngọc Bách
Sinh viên:	Nguyễn Việt Anh
Mã sinh viên:	B23DCCE009
Lớp:	D23CQCE06-B
Niên khoá:	2023-2028
Hệ đào tạo:	Đại học chính quy

Hà Nội, Tháng 5/2025

Mục lục

Giới thiệu tổng quan

Chương trình định nghĩa một hàm `cluster_players_with_kmeans_and_pca` chứa toàn bộ logic. Nó đọc dữ liệu cầu thủ từ một tệp CSV, làm sạch và chuẩn hóa dữ liệu, sử dụng phương pháp Elbow để hỗ trợ chọn số cụm (mặc dù sau đó chọn K=4 theo vai trò bóng đá), áp dụng K-means để phân nhóm, dùng PCA để giảm dữ liệu xuống 2 chiều, vẽ biểu đồ phân cụm 2D và cuối cùng, phân tích các đặc điểm trung bình của từng cụm và lưu kết quả phân tích vào tệp văn bản.

Giải thích chi tiết từng phần:

1.Nhập thư viện:

Python

```
import pandas as pd # Để làm việc với cấu trúc dữ liệu bảng
(DataFrame)
import numpy as np  # Để thực hiện các phép toán số học, xử lý mảng
from sklearn.preprocessing import StandardScaler # Để chuẩn hóa dữ
liệu (mean=0, variance=1)
from sklearn.cluster import KMeans # Thuật toán phân cụm K-means
from sklearn.impute import SimpleImputer # Để xử lý giá trị thiếu
(NaN)
from sklearn.decomposition import PCA # Phân tích thành phần chính
(giảm chiều)
import matplotlib.pyplot as plt # Để vẽ biểu đồ
import seaborn as sns # Thư viện nâng cao cho biểu đồ đẹp hơn, dễ sử
dụng hơn matplotlib
import os # Để làm việc với hệ thống file (tạo thư mục, kiểm tra
đường dẫn)
```

Đây là bước nhập các công cụ cần thiết từ các thư viện phổ biến trong Python cho khoa học dữ liệu và xử lý file.

2.Định nghĩa hàm cluster_players_with_kmeans_and_pca():

Python

```
def cluster_players_with_kmeans_and_pca():
    """
    Phân nhóm cầu thủ bằng K-means, giảm chiều bằng PCA, và vẽ biểu
đồ phân cụm 2D.
    """
    # ... (toàn bộ logic nằm trong hàm này)
```

Toàn bộ quy trình được gói gọn trong một hàm. Điều này giúp mã có cấu trúc, dễ đọc và dễ sử dụng lại. Docstring (chuỗi ký tự trong dấu `""" """`) giải thích mục đích của hàm.

3.Thiết lập đường dẫn file:

Python

```
input_csv_path = r"D:\python project\report\csv\result.csv"
output_analysis_path = r"D:\python
project\report\txt\kmeans_analysis_summary_k4.txt"
output_plot_dir = r"D:\python project\report\histograms\k-means"
elbow_plot_path = os.path.join(output_plot_dir,
"kmeans_elbow_plot.png")
pca_plot_path = os.path.join(output_plot_dir, "kmeans_pca_plot.png")
```

Các biến này lưu trữ đường dẫn đến tệp CSV đầu vào, tệp văn bản đầu ra cho bản tóm tắt phân tích và thư mục/tệp cho các biểu đồ. Sử dụng `r""` (chuỗi thô) giúp tránh lỗi với ký tự `\`. `os.path.join` là cách an toàn để kết hợp các phần của đường dẫn file, hoạt động tốt trên các hệ điều hành khác nhau.

4.Tải dữ liệu

Python

```
try:
    player_data_df = pd.read_csv(input_csv_path, na_values="N/A",
encoding="utf-8-sig")
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy tệp {input_csv_path}. Hãy đảm bảo
file tồn tại.")
    return
except Exception as e:
    print(f"Lỗi khi đọc tệp CSV: {e}")
    return

if player_data_df.empty:
    print("Tệp result.csv rỗng. Không có dữ liệu để phân tích.")
    return

print(f"Đã tải thành công {len(player_data_df)} cầu thủ từ
{input_csv_path}")
```

Đọc dữ liệu từ tệp CSV vào DataFrame của pandas.

- `na_values="N/A"`: Chỉ định rằng giá trị "N/A" trong tệp CSV nên được coi là giá trị thiếu (NaN).
- `encoding="utf-8-sig"`: Chỉ định mã hóa ký tự. "utf-8-sig" thường dùng cho tệp CSV được tạo bởi Microsoft Excel để xử lý Byte Order Mark (BOM).
- Khối `try...except` được sử dụng để xử lý các lỗi có thể xảy ra trong quá trình đọc file (ví dụ: file không tồn tại).
- Kiểm tra xem DataFrame có rỗng sau khi đọc hay không.

5.Tiền xử lý dữ liệu

- **Chọn cột thông tin:**

  Python

```
player_info_df = player_data_df[['Player', 'Team',
'Position']].copy()
```

Tạo một DataFrame riêng chỉ chứa thông tin định danh của cầu thủ. `.copy()` là quan trọng để tránh cảnh báo SettingWithCopyWarning sau này khi thêm cột 'Cluster'.

- **Chọn cột số để phân cụm:**

  Python

```
cols_to_exclude = ['Minutes', 'Matches Played', 'Starts']
features_for_clustering_df =
player_data_df.drop(columns=['Player', 'Nation', 'Team',
'Position'] + cols_to_exclude,

errors='ignore')
features_for_clustering_df =
features_for_clustering_df.select_dtypes(include=np.number)
```

  Loại bỏ các cột không cần thiết hoặc không phù hợp cho việc phân cụm dựa trên *chỉ số hiệu suất* (ví dụ: tên cầu thủ, đội, vị trí, quốc gia là thông tin phân loại; Cột `Minutes`, `Matches Played`, `Starts` bị loại bỏ vì chúng là tổng cộng dồn và ảnh hưởng lớn bởi thời gian thi đấu, thay vì phong cách chơi. Việc phân cụm thường hiệu quả hơn với các chỉ số 'per 90' hoặc các chỉ số không phụ thuộc vào thời gian chơi). `errors='ignore'` đảm bảo mã không báo lỗi nếu một cột trong danh sách loại trừ không tồn tại.
  `select_dtypes(include=np.number)` chỉ giữ lại các cột có kiểu dữ liệu số.

- **Kiểm tra cột sau lọc:**

  Python

```
if features_for_clustering_df.empty or
features_for_clustering_df.shape[1] == 0:
    print("Không có đủ cột chỉ số số học để thực hiện phân cụm
sau khi lọc.")
    return
print(f"Các chỉ số được sử dụng để phân cụm
({features_for_clustering_df.shape[1]} chỉ số):
{features_for_clustering_df.columns.tolist()}")
```

  Đảm bảo rằng vẫn còn các cột số để làm việc sau khi loại bỏ.

- **Xử lý giá trị thiếu:**

  Python

```
imputer = SimpleImputer(strategy='median')
imputed_features =
imputer.fit_transform(features_for_clustering_df)
imputed_features_df = pd.DataFrame(imputed_features,
columns=features_for_clustering_df.columns)
```

  `SimpleImputer` được sử dụng để điền vào các giá trị thiếu (NaN) trong các cột số. `strategy='median'` có nghĩa là các giá trị thiếu sẽ được thay thế bằng giá

trị trung vị (median) của cột đó. Trung vị ít bị ảnh hưởng bởi các giá trị ngoại lai hơn là trung bình. `fit_transform` vừa học (fit) trung vị từ dữ liệu vừa điền (transform).

- **Chuẩn hóa dữ liệu:**

  Python

  ```
  scaler = StandardScaler()
  scaled_features = scaler.fit_transform(imputed_features_df)
  scaled_features_df = pd.DataFrame(scaled_features,
  columns=imputed_features_df.columns)
  ```

  `StandardScaler` chuẩn hóa mỗi chỉ số sao cho nó có trung bình (mean) bằng 0 và độ lệch chuẩn (standard deviation) bằng 1. Điều này rất quan trọng đối với các thuật toán dựa trên khoảng cách như K-means, vì nó đảm bảo rằng các chỉ số có phạm vi giá trị lớn hơn không chi phối khoảng cách một cách không công bằng. `fit_transform` học các tham số chuẩn hóa (trung bình và độ lệch chuẩn) và áp dụng chúng.

6.Xác định số cụm tối ưu bằng phương pháp Elbow

Python

```
inertia_values = []
possible_k_values = range(2, 16)

for k in possible_k_values:
    kmeans_model = KMeans(n_clusters=k, init='k-means++',
n_init='auto', random_state=42, algorithm='lloyd')
    kmeans_model.fit(scaled_features_df)
    inertia_values.append(kmeans_model.inertia_)

# Vẽ và lưu biểu đồ Elbow
# ... (code vẽ biểu đồ)
```

Phương pháp Elbow chạy thuật toán K-means với một loạt các giá trị K khác nhau (ở đây là từ 2 đến 15). Đối với mỗi giá trị K, nó tính toán **Inertia**, là tổng bình phương khoảng cách từ mỗi điểm đến tâm cụm của nó. Inertia đo lường sự chặt chẽ của các cụm – giá trị thấp hơn nghĩa là các điểm gần tâm cụm hơn. Biểu đồ Inertia so với K thường có hình dạng giống cánh tay, và "khuỷu tay" (điểm mà sự giảm Inertia bắt đầu chậm lại đáng kể) được coi là gợi ý cho số cụm tối ưu.

- `init='k-means++'`: Phương pháp khởi tạo tâm cụm ban đầu thông minh hơn, giúp K-means hội tụ nhanh hơn và tránh được một số trường hợp kẹt vào cực tiểu cục bộ.
- `n_init='auto'`: Chạy thuật toán K-means nhiều lần với các tâm cụm ban đầu khác nhau và chọn kết quả tốt nhất (có Inertia thấp nhất). 'auto' tự động xác định số lần chạy.
- `random_state=42`: Đảm bảo kết quả có thể tái lập.
- `algorithm='lloyd'`: Thuật toán K-means cổ điển.

7.Áp dụng thuật toán K-means với K=4

Python

```
optimal_k = 4 # Chọn K=4 dựa trên vai trò bóng đá
final_kmeans_model = KMeans(n_clusters=optimal_k, init='k-means++',
n_init='auto', random_state=42,
                            algorithm='lloyd')
cluster_labels = final_kmeans_model.fit_predict(scaled_features_df)

# Thêm nhãn cụm vào DataFrame thông tin cầu thủ
player_info_df['Cluster'] = cluster_labels
```

Dựa trên quyết định (hoặc kết quả từ phương pháp Elbow), số cụm cuối cùng (`optimal_k`) được chọn là 4. Thuật toán K-means được chạy lại với K=4 trên dữ liệu đã chuẩn hóa. `fit_predict` vừa huấn luyện mô hình vừa trả về nhãn cụm cho mỗi điểm dữ liệu. Các nhãn này sau đó được thêm vào DataFrame chứa thông tin cầu thủ gốc.

8.Giảm chiều dữ liệu bằng PCA

Python

```
pca = PCA(n_components=2)
pca_features = pca.fit_transform(scaled_features_df)
explained_variance_ratio = pca.explained_variance_ratio_
print(f"Tỷ lệ phương sai được giải thích bởi 2 thành phần PCA:
{explained_variance_ratio.sum():.2%}")

# Tạo DataFrame cho dữ liệu PCA
pca_df = pd.DataFrame(data=pca_features, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = cluster_labels
pca_df['Player'] = player_info_df['Player'].values
pca_df['Position'] = player_info_df['Position'].values
```

`PCA` (Principal Component Analysis) là một kỹ thuật giảm chiều. Nó tìm ra các tổ hợp tuyến tính của các chỉ số gốc (gọi là các thành phần chính) nắm bắt được nhiều phương sai (thông tin, sự khác biệt) nhất trong dữ liệu.

- `n_components=2`: Giảm dữ liệu xuống còn 2 chiều (2 thành phần chính). Điều này là cần thiết để có thể vẽ biểu đồ 2D.
- `fit_transform(scaled_features_df)`: Áp dụng PCA lên dữ liệu *đã chuẩn hóa*.
- `explained_variance_ratio_`: Thuộc tính này cho biết tỷ lệ phương sai của dữ liệu gốc được giải thích bởi mỗi thành phần chính. Tổng của `explained_variance_ratio_` cho 2 thành phần đầu tiên cho biết tổng lượng thông tin được giữ lại sau khi giảm chiều xuống 2D.
- Một DataFrame mới (`pca_df`) được tạo ra để lưu trữ 2 thành phần PCA cùng với nhãn cụm, tên cầu thủ và vị trí gốc.

9.Vẽ biểu đồ phân cụm 2D

Python

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='Cluster',
palette='deep', style='Position', s=100)
# ... (code thêm tiêu đề, nhãn, chú giải, lưu và đóng biểu đồ)
```

Sử dụng thư viện `seaborn` để tạo biểu đồ phân tán (scatterplot).

- `data=pca_df`: Sử dụng DataFrame chứa dữ liệu PCA.
- `x='PCA1'`, `y='PCA2'`: Trục x và y là 2 thành phần chính.
- `hue='Cluster'`: Các điểm được tô màu dựa trên nhãn cụm mà K-means đã gán. Điều này giúp trực quan hóa sự tách biệt giữa các cụm.
- `palette='deep'`: Chọn bảng màu cho các cụm.
- `style='Position'`: Sử dụng các kiểu điểm đánh dấu khác nhau dựa trên vị trí *gốc* của cầu thủ. Điều này giúp đánh giá xem liệu các cụm được K-means tìm thấy có tương ứng với các vị trí chơi bóng đá truyền thống hay không.
- Biểu đồ này giúp chúng ta thấy các cụm có tách biệt tốt trong không gian 2 chiều hay không.

10.Phân tích và diễn giải các cụm

Python

```
analysis_df = imputed_features_df.copy()
analysis_df['Cluster'] = cluster_labels
analysis_df['Player'] = player_info_df['Player'].values
analysis_df['Position'] = player_info_df['Position'].values

cluster_summary_stats =
analysis_df.groupby('Cluster')[features_for_clustering_df.columns].me
an()
cluster_counts = analysis_df['Cluster'].value_counts().sort_index()

# ... (code tạo nội dung tóm tắt phân tích)
```

Phần này nhằm hiểu rõ hơn *ý nghĩa* của từng cụm.

- Một DataFrame mới (`analysis_df`) được tạo ra, bao gồm các chỉ số *gốc* (sau khi xử lý NaN nhưng *trước* khi chuẩn hóa), cùng với nhãn cụm, tên cầu thủ và vị trí. Việc sử dụng dữ liệu gốc giúp việc diễn giải các giá trị trung bình dễ dàng hơn (ví dụ: "trung bình 5 bàn thắng" dễ hiểu hơn "trung bình 1.5 trên thang điểm chuẩn hóa").
- `groupby('Cluster').mean()`: Tính giá trị trung bình của *tất cả* các chỉ số được sử dụng để phân cụm, cho mỗi cụm. Bằng cách so sánh các giá trị trung bình này giữa các cụm, chúng ta có thể xác định những chỉ số nào là đặc trưng cho từng cụm (ví dụ: Cụm A có trung bình số bàn thắng cao, trong khi Cụm B có trung bình số đường chuyền chính xác cao).
- `value_counts()`: Đếm số lượng cầu thủ trong mỗi cụm.
- Phần còn lại của khối này xây dựng nội dung cho tệp phân tích đầu ra dưới dạng một danh sách các chuỗi. Nó bao gồm số lượng cầu thủ trong mỗi cụm, bảng các chỉ số trung bình cho từng cụm, ví dụ về các cầu thủ trong mỗi cụm và các nhận xét giải thích kết quả (dựa trên số cụm đã chọn, sự phù hợp của các cụm với vị trí gốc, và nhận xét về biểu đồ PCA). Các nhận xét này là do

người viết mã đưa ra để diễn giải ý nghĩa của các cụm dựa trên các chỉ số trung bình và vị trí cầu thủ ví dụ.

11.Lưu kết quả phân tích

Python

```
try:
    os.makedirs(os.path.dirname(output_analysis_path), exist_ok=True)
    with open(output_analysis_path, 'w', encoding='utf-8') as f:
        f.writelines(analysis_output_content)
    print(f"Đã lưu tóm tắt phân tích cụm vào:
{output_analysis_path}")
except Exception as e:
    print(f"Lỗi khi lưu tệp kết quả: {e}")
```

Lưu nội dung phân tích đã tạo ở bước trước vào tệp văn bản được chỉ định.

- `os.makedirs(..., exist_ok=True)`: Tạo thư mục chứa tệp đầu ra nếu nó chưa tồn tại. `exist_ok=True` ngăn lỗi nếu thư mục đã có.
- Sử dụng `with open(...)` đảm bảo tệp được đóng đúng cách sau khi ghi.
- `'w'` để ghi (sẽ ghi đè nếu tệp đã tồn tại).
- `encoding='utf-8'` để đảm bảo các ký tự tiếng Việt được lưu đúng.
- `writelines()` ghi danh sách các chuỗi vào tệp.
- Thêm khối `try...except` để xử lý lỗi khi lưu file.

12.Khối if __name__ == "__main__"::

Python

```
if __name__ == "__main__":
    cluster_players_with_kmeans_and_pca()
```

Đây là một cấu trúc Python tiêu chuẩn. Nó đảm bảo rằng hàm `cluster_players_with_kmeans_and_pca()` chỉ được gọi khi script được chạy trực tiếp (ví dụ: `python ten_file_script.py`), chứ không phải khi nó được nhập khẩu như một module trong một script khác.

Tóm lại:

Chương trình này là một ví dụ điển hình về cách áp dụng các kỹ thuật học máy (Machine Learning) cơ bản (phân cụm K-means, giảm chiều PCA) cho dữ liệu thực tế. Quy trình từ tải dữ liệu, tiền xử lý, mô hình hóa đến trực quan hóa và phân tích được thực hiện một cách có hệ thống. Việc chọn K=4 dựa trên kiến thức chuyên môn (vai trò trong bóng đá) là một quyết định thiết kế thú vị, cho thấy sự kết hợp giữa phân tích dữ liệu và hiểu biết về miền dữ liệu. Biểu đồ PCA và bảng thống kê trung bình cụm là những công cụ hữu ích để diễn giải kết quả phân cụm, mặc dù cần lưu ý rằng PCA chỉ giữ lại một phần phương sai của dữ liệu gốc.