Buổi thực hành 5

Phân cụm dữ liệu với K-Means

1. Mục tiêu và Liên kết môn học:

- **Mục tiêu:**
  - Sinh viên triển khai thành thạo thuật toán K-Means và các phương pháp đánh giá (Elbow, Silhouette).
  - Sinh viên thực hiện được các bước tiền xử lý dữ liệu thực tế: xử lý giá trị thiếu (Missing Value Imputation) và chuẩn hóa dữ liệu (Standardization).
  - Sinh viên hiểu rõ **lý do tại sao** phải chuẩn hóa dữ liệu khi làm việc với các đặc trưng có thang đo khác nhau
  - Sinh viên có khả năng **phân tích và diễn giải ý nghĩa nghiệp vụ** của các cụm được hình thành trong không gian **nhiều chiều** (nơi không thể trực quan hóa 2D).
- **Chuẩn đầu ra (CLOs) đáp ứng:**
  - **CLO2:** Vận dụng được quy trình thực hiện của một hệ thống máy học ứng dụng (thu thập, **tiền xử lý dữ liệu (xử lý thiếu, chuẩn hóa)**, lựa chọn mô hình, huấn luyện và đánh giá mô hình).
  - **CLO3:** Phân tích, đánh giá và lựa chọn được các mô hình máy học phù hợp với yêu cầu của bài toán (chọn k cho K-Means).
  - **CLO4:** Sử dụng thành thạo các công cụ, thư viện máy học (Pandas, Scikit-learn: SimpleImputer, StandardScaler, KMeans) để giải quyết các bài toán thực tế.

2. Công cụ và Dữ liệu:

- **Công cụ:** Python, Jupyter Notebook/Google Colab, và các thư viện: Pandas, Scikit-learn (SimpleImputer, StandardScaler, KMeans), Matplotlib, Seaborn.
- **Dữ liệu:**
  - **Phần 1 (Code-along):** Bộ dữ liệu "Mall Customers" (Khách hàng trung tâm thương mại).
    - *Mục đích:* Dùng làm ví dụ cơ bản, trực quan (2D) để hiểu K-Means, Elbow, Silhouette và tầm quan trọng của chuẩn hóa.
  - **Phần 2 (Tự làm):** Bộ dữ liệu "Credit Card Dataset (Phân khúc chủ thẻ tín dụng)".
    - *Mô tả:* Dữ liệu hành vi của ~9000 chủ thẻ tín dụng với 17 đặc trưng.
    - *Thách thức (Mới):* Dữ liệu có **giá trị bị thiếu** và **nhiều chiều (16 features)**.

Sinh viên phải phân cụm trên 16 chiều và diễn giải kết quả mà không cần visualization.

- *Nguồn:* Cung cấp file CC_GENERAL.csv cho sinh viên.

3. Nội dung thực hành:

Phần 1: Hướng dẫn thực hành (Code-along - 150 phút)

- **Bài toán:** Phân nhóm khách hàng "Mall Customers" (2 đặc trưng).

Bước 1: Tải và Khám phá dữ liệu (EDA)

- **Mục đích:** Hiểu cấu trúc dữ liệu, kiểm tra dữ liệu thiếu và chọn các đặc trưng cần thiết.

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import warnings
warnings.filterwarnings('ignore') # Tắt bớt cảnh báo (tùy chọn)

# Tải dữ liệu (Giả sử có file 'Mall_Customers.csv')
data = pd.read_csv('Mall_Customers.csv')
# Chọn các cột liên quan cho việc phân cụm
X = data.iloc[:, [3, 4]].values

# Trực quan hóa dữ liệu để xem sơ bộ
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1])
plt.title('Phân bố khách hàng (Chưa phân cụm)')
plt.xlabel('Thu nhập hàng năm (k$)')
plt.ylabel('Điểm chi tiêu (1-100)')
plt.show()
```

- **Diễn giải:** Dữ liệu tải thành công. Biểu đồ 2D cho thấy dữ liệu có vẻ tụ lại thành 5 nhóm.

Bước 2: Tiền xử lý - Chuẩn hóa dữ liệu

- **Mục đích (Giải thích lý do):** Thuật toán K-Means dựa trên khoảng cách Euclidean. Trong dữ liệu này, Thu nhập hàng năm (thang đo 15k-137k) và Điểm chi tiêu (thang đo 1-99) có thang đo (scale) rất khác nhau.
    - **Vấn đề:** Nếu không chuẩn hóa, sự chênh lệch 1k$ (ví dụ: từ 50k lên 51k) sẽ được K-Means "hiểu" là lớn hơn rất nhiều so với sự chênh lệch 1 điểm chi tiêu (ví dụ: từ 50 lên 51). Đặc trưng Thu nhập sẽ "lấn át" hoàn toàn đặc trưng Điểm chi tiêu.
    - **Giải pháp:** Chúng ta dùng StandardScaler (như trong BaiGiang_MayHocUngDung_C2.pdf, slide 133) để đưa cả hai đặc trưng về cùng một thang đo (trung bình = 0, độ lệch chuẩn = 1). Điều này đảm bảo mỗi đặc trưng đều có "trọng số" như nhau trong việc tính toán khoảng cách.

Python

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("\nDữ liệu sau khi chuẩn hóa (5 dòng đầu):")
print(X_scaled[:5])
```

- **Diễn giải:** Dữ liệu đã được chuẩn hóa. Mô hình K-Means sẽ được huấn luyện trên X_scaled.

Bước 3: Xác định số cụm tối ưu (k) bằng Phương pháp Elbow & Silhouette

- **Mục đích:** K-Means yêu cầu chúng ta phải chỉ định trước số cụm (k). Chúng ta dùng 2 phương pháp để tìm k "tốt" nhất:
    - **Elbow:** Tìm điểm "khuỷu tay" - là điểm mà sau đó, việc tăng k không còn giúp giảm WCSS (tổng bình phương sai số trong cụm) một cách đáng kể nữa.
    - **Silhouette:** Tìm k cho giá trị Silhouette trung bình cao nhất (càng gần 1, cụm càng rõ ràng và tách biệt).

Python

```
# Chạy Elbow
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++',
max_iter=300, n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_) # inertia_ chính là WCSS

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
```

```
plt.title('Phương pháp Elbow')
plt.xlabel('Số lượng cụm (k)')
plt.ylabel('WCSS (Inertia)')
plt.grid(True)

# Chạy Silhouette
silhouette_scores = []
for k in range(2, 11): # Silhouette yêu cầu ít nhất 2 cụm
    kmeans_iter = KMeans(n_clusters=k, init='k-means++',
n_init=10, random_state=42)
    kmeans_iter.fit(X_scaled)
    score = silhouette_score(X_scaled,
kmeans_iter.labels_)
    silhouette_scores.append(score)

plt.subplot(1, 2, 2)
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Silhouette Score cho các giá trị k')
plt.xlabel('Số lượng cụm (k)')
plt.ylabel('Silhouette Score')
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Diễn giải:** Cả hai biểu đồ đều chỉ ra **k = 5** là lựa chọn tối ưu.

Bước 4: Huấn luyện mô hình K-Means với k=5 và Trực quan hóa

- **Mục đích:** Áp dụng K-Means với k tối ưu đã chọn (k=5) và xem kết quả trực quan.

Python

```
k_optimal = 5
kmeans_model = KMeans(n_clusters=k_optimal, init='k-
means++', max_iter=300, n_init=10, random_state=42)
y_kmeans = kmeans_model.fit_predict(X_scaled)

# Thêm nhãn cụm vào DataFrame gốc để tiện phân tích
data['Cluster'] = y_kmeans

# Trực quan hóa
plt.figure(figsize=(10, 8))
sns.scatterplot(data=data, x='Annual Income (k$)',
```

```
y='Spending Score (1-100)',
                hue='Cluster', palette='Set1', s=60)
plt.scatter(scaler.inverse_transform(kmeans_model.cluster_
centers_)[:,0],

scaler.inverse_transform(kmeans_model.cluster_centers_)[:,
1],
            s=200, c='yellow', marker='*', label='Tâm
cụm')
plt.title('Phân cụm khách hàng (K=5)')
plt.legend()
plt.grid(True)
plt.show()
```

- **Diễn giải:** Biểu đồ hiển thị 5 cụm khách hàng. Chúng ta có thể dễ dàng diễn giải bằng mắt thường:
  - Cụm 0 (ví dụ): Thu nhập trung bình, chi tiêu trung bình.
  - Cụm 1: Thu nhập cao, chi tiêu thấp (nhóm "Cẩn trọng").
  - Cụm 2: Thu nhập thấp, chi tiêu thấp.
  - Cụm 3: Thu nhập thấp, chi tiêu cao (nhóm "Tiềm năng rủi ro").
  - Cụm 4: Thu nhập cao, chi tiêu cao (nhóm "VIP/Target").

Phần 2: Bài tập tự làm và nộp bài

- **Bài toán:** Phân khúc các chủ thẻ tín dụng (dữ liệu CC_GENERAL.csv).
- **Mục tiêu:** Sinh viên tự mình xử lý một bài toán phức tạp hơn: dữ liệu bị thiếu và **dữ liệu 16 chiều**. Thách thức ở đây là sinh viên **không thể trực quan hóa 16 chiều** mà phải dựa vào phân tích số liệu.

Yêu cầu (Tasks):

1. **Tải và Khám phá dữ liệu:**
   - **Công việc:** Tải dữ liệu, dùng df.info() và df.isnull().sum(). Bỏ cột CUST_ID.
   - **Giải thích (Tại sao?):**
     - df.isnull().sum(): Để biết chính xác cột nào bị thiếu dữ liệu. Các thuật toán máy học sẽ báo lỗi nếu nhận vào giá trị NaN (Not a Number).
     - Bỏ CUST_ID: Cột này là mã định danh, nó không phải là một "hành vi" hay "đặc trưng" (feature). Giống như số CMND không nói lên bạn là người như thế nào. Giữ lại cột này sẽ gây nhiễu và làm hỏng kết quả phân cụm.

2. **Tiền xử lý (Dữ liệu thiếu):**
    - **Công việc:** Sử dụng SimpleImputer (từ sklearn.impute) hoặc df.fillna() để điền vào các giá trị bị thiếu (ở cột CREDIT_LIMIT và MINIMUM_PAYMENTS).
3. **Chuẩn hóa dữ liệu (Thách thức 2: Dữ liệu 16 chiều):**
    - **Công việc:** Sử dụng StandardScaler để chuẩn hóa tất cả 16 đặc trưng còn lại.
4. **Tìm k tối ưu:**
    - **Công việc:** Chạy phương pháp Elbow và Silhouette trên **toàn bộ 16 đặc trưng đã chuẩn hóa** (X_scaled_16D).
    - **Giải thích:** Tương tự Phần 1, chúng ta cần tìm số cụm k tối ưu cho dữ liệu 16 chiều này. Quá trình này có thể mất nhiều thời gian hơn một chút. (Gợi ý: Kết quả có thể không rõ ràng như 2D, sinh viên cần đưa ra lựa chọn và *biện minh* cho nó, ví dụ: "Em chọn k=6 vì Silhouette cao nhất", hoặc "Elbow không rõ, nhưng Silhouette cho thấy k=5 là tốt").
5. **Huấn luyện K-Means:**
    - **Công việc:** Chọn giá trị k tối ưu (ví dụ k=6), huấn luyện K-Means trên dữ liệu 16 chiều đã chuẩn hóa.
6. **Phân tích & Diễn giải (Phần quan trọng nhất - Thách thức 3):**
    - **Công việc:**
        1. Tạo một DataFrame mới, copy từ DataFrame đã chuẩn hóa ở Bước 3 (có 16 đặc trưng).
        2. Thêm cột 'Cluster' (chứa nhãn y_pred từ Bước 5) vào DataFrame này.
        3. Thực hiện df_analyis.groupby('Cluster').mean() để tính giá trị trung bình của **16 đặc trưng gốc** (đã chuẩn hóa) cho mỗi cụm.
7. **Phân tích so sánh (Câu hỏi tư duy):**
    - **Công việc:**
        1. Bây giờ, hãy thử làm lại một phân tích *đơn giản*: chỉ chọn 2 đặc trưng bạn cho là quan trọng nhất (ví dụ: BALANCE và PURCHASES).
        2. Chuẩn hóa 2 đặc trưng này.
        3. Chạy K-Means (với cùng k) trên chỉ 2 đặc trưng này.
        4. Vẽ biểu đồ 2D của chúng.
    - **Câu hỏi:**
        - Các cụm trong biểu đồ 2D này có "rõ ràng" như ở Phần 1 không?
        - Bạn có nghĩ rằng việc chỉ dùng 2 đặc trưng (BALANCE và PURCHASES) đã bỏ lỡ nhiều thông tin quan trọng từ 14 đặc trưng còn lại (như

CASH_ADVANCE, PURCHASES_FREQUENCY...) không?

- Theo bạn, kết quả phân cụm 16 chiều (ở Bước 6) hay kết quả 2D (ở Bước 7) đáng tin cậy hơn để phân khúc khách hàng? Tại sao?