mini-project-3

August 9, 2025

Ứng dụng thuật toán học máy để phân nhóm khu vực nhà ở theo điều kiện kinh tế

Nhóm 9:

- Nguyễn Ngọc Hải Haui-2022605947
- Nguyễn Thành Công Haui-2022606702
- Vũ Minh Đức 11221425
- Nguyễn Hoàng Nguyên 11224818
- Nguyễn Trọng Vỹ 11227025

1 Giới thiệu bài toán

Dataset này được sử dụng trong Chương 2 của cuốn sách nổi tiếng "Hands-On Machine Learning with Scikit-Learn and TensorFlow" của Aurélien Géron. Đây là bộ dữ liệu lý tưởng dành cho người mới học về Machine Learning do có các đặc điểm:

Cấu trúc đơn giản, dễ hiểu.

Dữ liệu thực tế từ tổng điều tra dân số California năm 1990.

Không quá lớn để xử lý, cũng không quá nhỏ để không có ý nghĩa.

Dữ liệu cung cấp thông tin về các khu dân cư tại California và các thống kê liên quan đến nhà ở và dân số tại từng khu vực nhỏ (gọi là district).

- Features của dataset:
  - longitude: Kinh độ của khu vực (tọa độ địa lý, theo hướng Đông/Tây).
  - latitude: Vĩ độ của khu vực (tọa độ địa lý, theo hướng Bắc/Nam).
  - housing_median_age: Tuổi trung vị của các căn nhà trong khu vực .
  - total_rooms: Tổng số phòng của tất cả các căn nhà trong khu vực.
  - total_bedrooms: Tổng số phòng ngủ trong tất cả các căn nhà.
  - population: Tổng dân số đang sinh sống trong khu vực.
  - households: Tổng số hộ gia đình trong khu vực.
  - median_income: Thu nhập trung vị của hộ dân trong khu vực.
  - median_house_value: Giá nhà trung vị tại khu vực (USD).

– ocean_proximity: Vị trí tương đối so với biển.

Nguồn dữ liệu:https://www.kaggle.com/datasets/camnugent/california-housing-prices/data

Mục tiêu: Phân vùng kinh tế (Giàu, nghèo, trung bình)

Dữ liệu có 20640 dòng dữ liệu và đều là biến numerical trừ 1 biến categoterical

Kết quả mong đợi của báo cáo tìm ra thuật toán phù hợp với bài toán 1 trong 5 thuật toán:

- DBSCAN
- Gaussian Mixture
- Spectral Clustering
- Agglomerative Clustering
- Kmeans

2 Import thư viện

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from ydata_profiling import ProfileReport
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.mixture import GaussianMixture
from sklearn.cluster import SpectralClustering
from sklearn.cluster import AgglomerativeClustering
from sklearn.model_selection import GridSearchCV
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
```

3 Đọc và khám phá dữ liệu

[ ]:
```
df = pd.read_csv('housing.csv/housing.csv')
df.head()
```

[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41.0	880.0	129.0	
1	-122.22	37.86	21.0	7099.0	1106.0	
2	-122.24	37.85	52.0	1467.0	190.0	
3	-122.25	37.85	52.0	1274.0	235.0	
4	-122.25	37.85	52.0	1627.0	280.0	

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

[3]:
```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

[3]:

	longitude	latitude	housing_median_age	total_rooms	\
count	20640.000000	20640.000000	20640.000000	20640.000000	
mean	-119.569704	35.631861	28.639486	2635.763081	
std	2.003532	2.135952	12.585558	2181.615252	
min	-124.350000	32.540000	1.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1447.750000	
50%	-118.490000	34.260000	29.000000	2127.000000	
75%	-118.010000	37.710000	37.000000	3148.000000	
max	-114.310000	41.950000	52.000000	39320.000000	

	total_bedrooms	population	households	median_income	\
count	20433.000000	20640.000000	20640.000000	20640.000000	
mean	537.870553	1425.476744	499.539680	3.870671	
std	421.385070	1132.462122	382.329753	1.899822	
min	1.000000	3.000000	1.000000	0.499900	
25%	296.000000	787.000000	280.000000	2.563400	
50%	435.000000	1166.000000	409.000000	3.534800	
75%	647.000000	1725.000000	605.000000	4.743250	
max	6445.000000	35682.000000	6082.000000	15.000100	

```
       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

[4]:
```
print(df.isnull().sum())
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

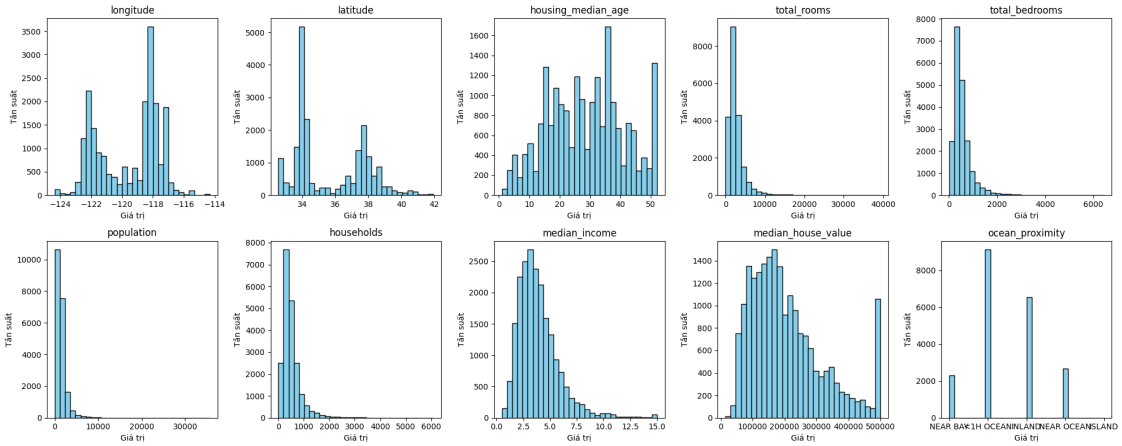
4 Trực quan hóa dữ liệu

[10]:
```
report = ProfileReport(df,title ="Profiling Report")
report.to_file("report.html")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
100%|      | 10/10 [00:00<00:00, 31.33it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

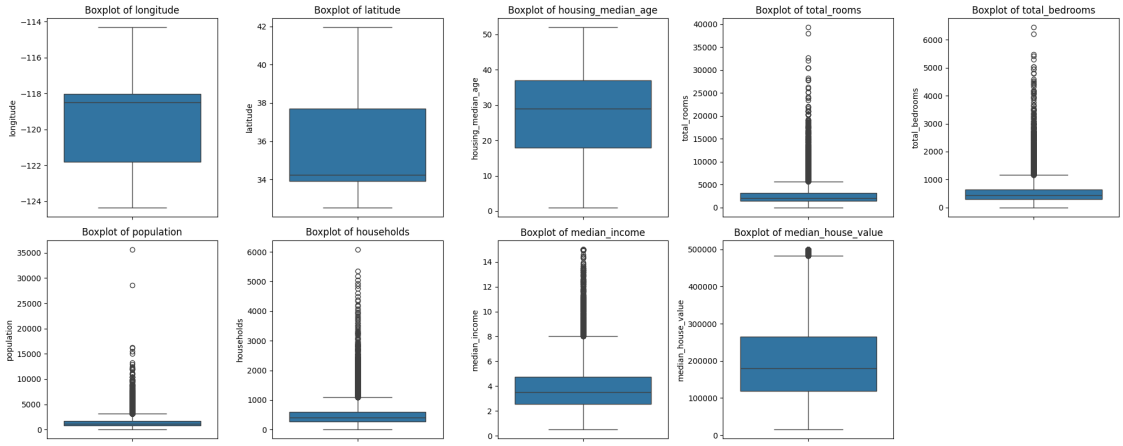
[11]:
```
# Chọn 10 đặc trưng đầu tiên
features_to_plot = df.columns[:10]

# Thiết lập figure 2 hàng 5 cột
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 8))
axes = axes.flatten()

# Vẽ histogram
for i, col in enumerate(features_to_plot):
```

```
    axes[i].hist(df[col], bins=30, color='skyblue', edgecolor='black')
    axes[i].set_title(col)
    axes[i].set_xlabel("Giá trị")
    axes[i].set_ylabel("Tần suất")

plt.tight_layout()
plt.show()
```



Dữ liệu phân bố không đồng đều thiên về lệnh về bên trái ảnh hưởng đến khi phân cụm theo kinh tế

[12]:
```
features = df.select_dtypes(include=['float64', 'int64']).columns[:10]

# Thiết lập kích thước biểu đồ
plt.figure(figsize=(20, 8))

# Vẽ 10 boxplots trên lưới 2 hàng 5 cột
for i, col in enumerate(features):
    plt.subplot(2, 5, i + 1)
    sns.boxplot(y=df[col])
    plt.title(f"Boxplot of {col}")

plt.tight_layout()
plt.show()
```



Biến có nhiều outlier: bedroom, toilet, floors

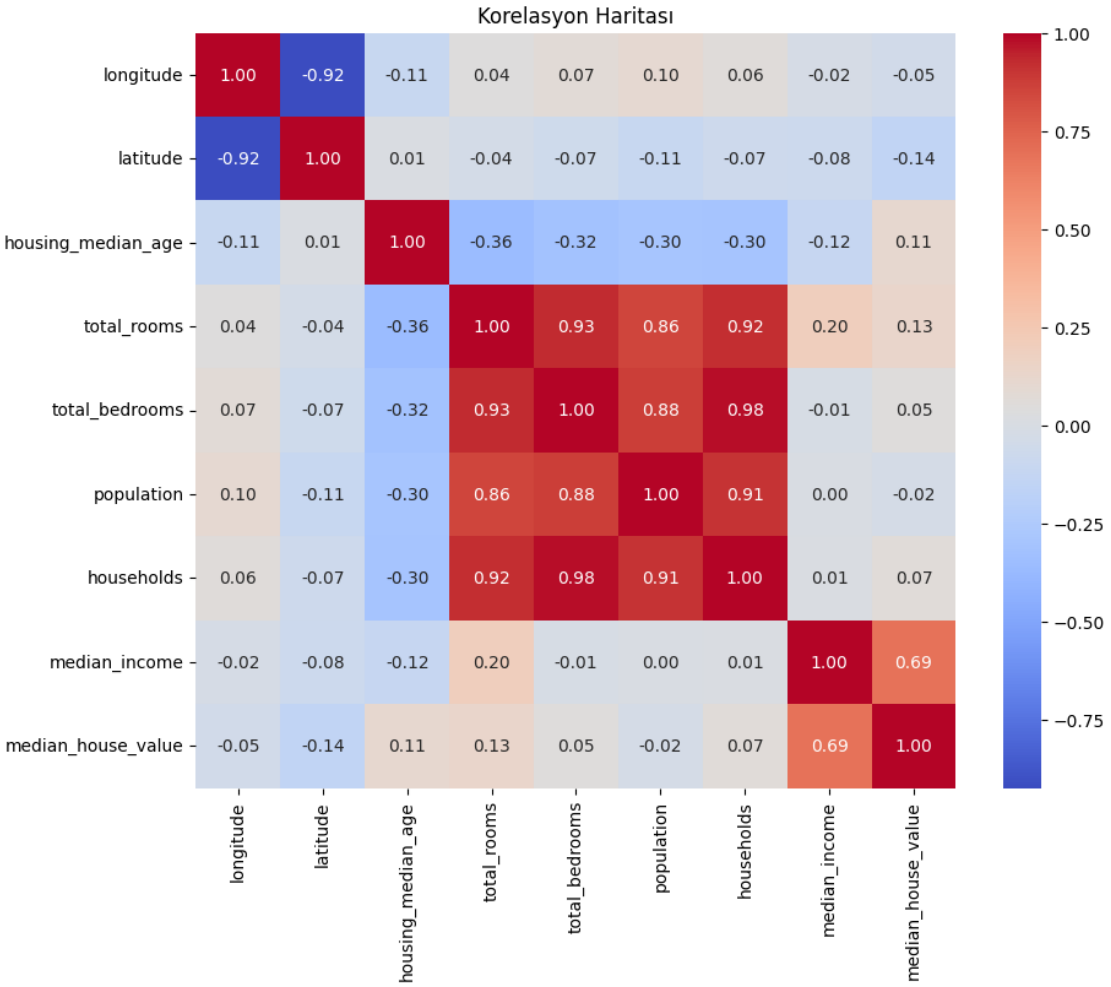
- Có rất nhiều dấu chấm nhỏ nằm ngoài phần thân boxplot, biểu thị sự hiện diện nhiều giá trị ngoại lệ.
- Những ngoại lệ này thường là các giá trị rất cao, có thể do lỗi nhập liệu hoặc những bất động sản hiếm (biệt thự, cao ốc…).

Biến có ít outlier: area

- Có một vài dấu chấm nhỏ nằm ngoài box, cho thấy có ít ngoại lệ hơn so với 3 biến trên.
- Đây có thể là một vài căn hộ/nhà đặc biệt lớn.

=> cần xử lí trước khi cho vào model

[8]:
```
corr_matrix = df.corr(numeric_only=True)

# Görselleştir
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Korelasyon Haritası")
plt.show()
```



Tương quan dương mạnh:

- median_income median_house_value: +0.69 → Thu nhập cao thì giá nhà cao.
- total_rooms total_bedrooms, households, population: từ +0.86 đến +0.98 → Phản ánh mối liên hệ chặt chẽ giữa quy mô nhà và dân số.

Tương quan âm mạnh:

- longitude latitude: -0.92 → Tương quan địa lý mạnh nhưng không hữu ích phân cho bài toán.

Tương quan thấp với median_house_value:

- Các biến như population, total_rooms, total_bedrooms có hệ số < 0.15 → Không đóng vai trò lớn việc phân cụm.

5 Tiền xử lý dữ liệu

Để phân cụm thành 3 vùng giàu - trung bình - nghèo, ta cần chọn các đặc trưng (feature) phản ánh điều kiện sống, tài chính hoặc tiện nghi của một khu vực.

[5]:
```
df.dropna(inplace=True)
features = ['median_income', 'median_house_value',
            'housing_median_age', 'total_rooms',
            'total_bedrooms', 'population', 'households']

X = df[features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

6 DBSCAN(Nguyễn Thành Công)

6.1 Lý do lựa chọn

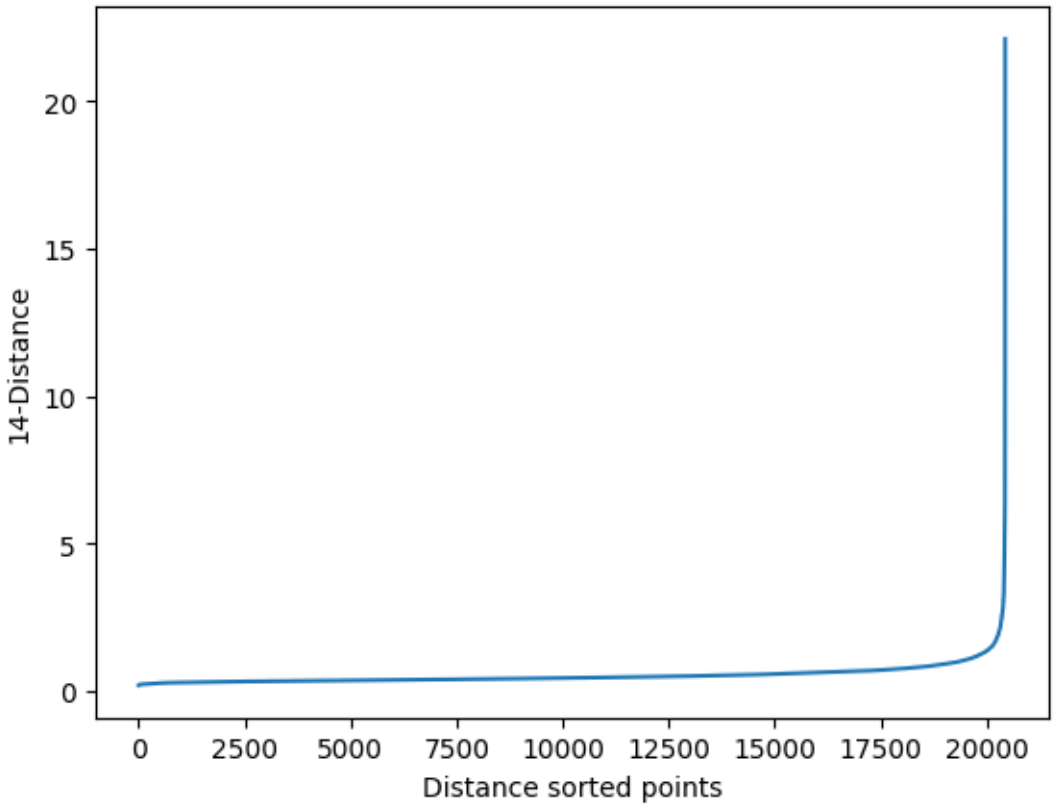
- Phân cụm theo mật độ: Dữ liệu nhà ở tập trung ở các thành phố và thưa thớt ở vùng nông thôn. DBSCAN rất hiệu quả trong việc tìm ra các khu vực có mật độ cao này, phản ánh đúng các khu dân cư thực tế.
- Xử lý hình dạng bất kỳ: DBSCAN không giả định các cụm có hình tròn (như K-Means). Nó có thể phát hiện các cụm có hình dạng phức tạp, ví dụ như các khu - dân cư trải dài dọc theo bờ biển hoặc đường cao tốc.
- Phát hiện dữ liệu nhiễu (outliers): DBSCAN có khả năng xác định và bỏ qua các căn nhà "đặc biệt" hoặc biệt lập, không gán chúng vào bất kỳ cụm nào. Điều này giúp các cụm còn lại trở nên ý nghĩa và chính xác hơn.
- Không cần xác định số cụm: Bạn không cần phải biết trước có bao nhiêu cụm khu dân cư. DBSCAN sẽ tự động tìm số lượng cụm dựa trên các tham

6.2 Tìm eps tối ưu

[6]:
```
k = 14
nbrs = NearestNeighbors(n_neighbors=k+1).fit(X_scaled)
dist, ind = nbrs.kneighbors(X_scaled)
```

[7]:
```
k_dist = np.sort(dist[:, -1])
```

[8]:
```
plt.plot(k_dist)
plt.xlabel('Distance sorted points')
plt.ylabel(f'{k}-Distance')
plt.show()
```



[ ]:
```python
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
import numpy as np

def find_best_dbscan_with_exact_clusters(X_scaled, eps_range,␣
 ↪min_samples_range, desired_clusters=3):
    best_score = -1
    best_params = None
    results = []

    for eps in eps_range:
        for min_samples in min_samples_range:
            db = DBSCAN(eps=eps, min_samples=min_samples)
            labels = db.fit_predict(X_scaled)

            # Bỏ nhiễu
            mask = labels != -1
            core_labels = labels[mask]
            num_clusters = len(set(core_labels))

            # Chỉ chấp nhận nếu có đúng số cụm mong muốn =3
```

```
            if num_clusters != desired_clusters:
                continue

            # Tính silhouette
            try:
                score = silhouette_score(X_scaled[mask], core_labels)
                results.append((eps, min_samples, score))
                if score > best_score:
                    best_score = score
                    best_params = (eps, min_samples)
            except:
                continue  # Tránh lỗi nếu không đủ điểm

    return best_params, best_score, results
eps_range = np.arange(0.1, 1.2, 0.1)
min_samples_range = range(3, 21)

best_params, best_score, results =␣
 ↪find_best_dbscan_with_exact_clusters(X_scaled, eps_range, min_samples_range)

if best_params:
    print(f" Tìm được cấu hình tốt nhất:")
    print(f"→ eps = {best_params[0]:.2f}, min_samples = {best_params[1]}")
    print(f"→ Silhouette Score = {best_score:.4f}")
else:
    print(" Không tìm được cấu hình nào tạo đúng 3 cụm.")
```

[ ]:
```
eps_range = np.arange(0.1, 1.2, 0.1)
min_samples_range = range(3, 21)

best_params, best_score, results =␣
 ↪find_best_dbscan_with_exact_clusters(X_scaled, eps_range, min_samples_range)

if best_params:
    print(f" Tìm được cấu hình tốt nhất:")
    print(f"→ eps = {best_params[0]:.2f}, min_samples = {best_params[1]}")
    print(f"→ Silhouette Score = {best_score:.4f}")
else:
    print(" Không tìm được cấu hình nào tạo đúng 3 cụm.")
```

Tìm được cấu hình tốt nhất: → eps = 1.00, min_samples = 5 → Silhouette Score = 0.7164

6.3 Huấn luyện mô hình

[9]:
```
epsilon = 1
minimumSamples = 5
dbDBSCAN = DBSCAN(eps=epsilon, min_samples=minimumSamples).fit(X_scaled)
labels = dbDBSCAN.labels_
labels
```

[9]:
```
array([0, 0, 0, …, 0, 0, 0], shape=(20433,))
```

[6]:
```
# Số cụm của mô hình
np.unique(labels)
```

[6]:
```
array([-1,  0,  1,  2], dtype=int64)
```

[7]:
```
num_noise = np.sum(labels == -1)
print(f"Số điểm nhiễu: {num_noise}")
```

```
Số điểm nhiễu: 325
```

[8]:
```
# Ngoại lai = False, cụm = True
core_samples_mask = np.zeros_like(dbDBSCAN.labels_, dtype=bool)
core_samples_mask[dbDBSCAN.core_sample_indices_] = True
core_samples_mask
```

[8]:
```
array([ True,  True,  True, …,  True,  True,  True])
```

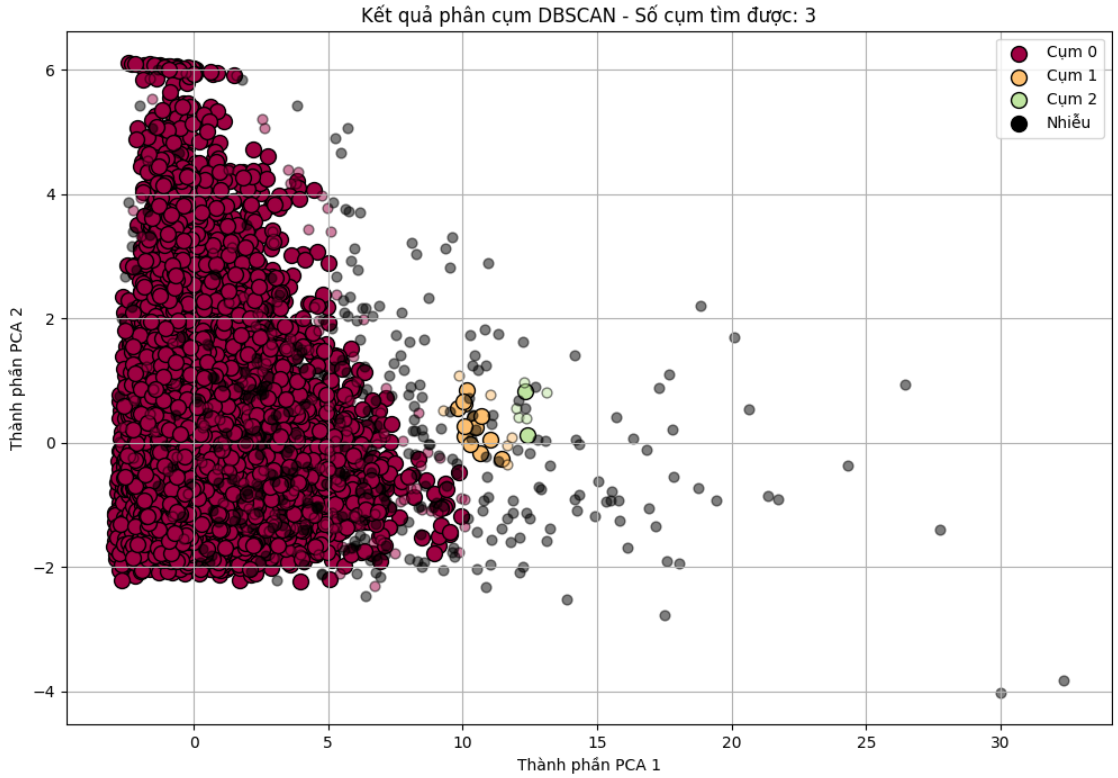
[9]:
```
#Số cụm thực tế
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_clusters_
```

[9]:
```
3
```

6.4 Trực quan hoá và phân tích dữ liệu

[11]:
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

unique_labels = set(labels)
n_clusters = len(unique_labels) - (1 if -1 in labels else 0)
colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))

plt.figure(figsize=(12, 8))

for k, col in zip(unique_labels, colors):
    if k == -1:
        # Màu đen cho các điểm nhiễu
```

```
        col = 'k'

    class_member_mask = (labels == k)

    # Vẽ các điểm lõi
    xy_core = X_pca[class_member_mask & core_samples_mask]
    plt.scatter(
        xy_core[:, 0], xy_core[:, 1],
        s=100, c=[col], marker='o',
        edgecolor='k', label=f'Cụm {k}' if k != -1 else 'Nhiễu'
    )

    # Vẽ các điểm biên
    xy_border = X_pca[class_member_mask & ~core_samples_mask]
    plt.scatter(
        xy_border[:, 0], xy_border[:, 1],
        s=40, c=[col], marker='o',
        edgecolor='k', alpha=0.5
    )

plt.title(f"Kết quả phân cụm DBSCAN - Số cụm tìm được: {n_clusters}")
plt.xlabel('Thành phần PCA 1')
plt.ylabel('Thành phần PCA 2')
plt.legend()
plt.grid(True)
plt.show()
```



[10]:
```
dfDBSCAN = df.copy()
dfDBSCAN['cluster'] = labels
dfDBSCAN[dfDBSCAN['cluster'] != -1].groupby('cluster')[[
    'median_income', 'median_house_value',
    'housing_median_age', 'total_rooms',
    'total_bedrooms', 'population', 'households'
]].mean().round(2)
```

[10]:

cluster	median_income	median_house_value	housing_median_age	total_rooms	total_bedrooms	population	households
0	3.86	205953.35	28.83	2487.37	508.67	1354.72	475.31
1	5.72	245800.00	7.68	15794.89	2507.00	7369.00	2358.74
2	6.21	284571.43	10.14	17483.00	2855.00	8644.86	2746.43

Cụm	Nhóm	Số lượng điểm	Đặc điểm nổi bật
0	**Nghèo**	20,082	Có mức thu nhập và giá nhà thấp hơn đáng kể so với hai cụm còn lại. Số lượng phòng và dân cư cũng thấp. Đây là cụm chính chiếm đa số dữ liệu.
1	**Trung bình**	19	Có mức thu nhập và giá nhà cao hơn cụm 0. Số phòng và dân cư ở mức lớn. Có thể đại diện cho các vùng khá giả.
2	**Giàu**	7	Có mức thu nhập trung vị, giá nhà, và quy mô nhà ở/dân cư cao nhất. Đây là nhóm đặc biệt, đại diện cho vùng dân cư giàu có nhất.
-1	**Nhiễu**	325	Không thuộc cụm nào do đặc điểm không đồng nhất hoặc nằm ở rìa các cụm. Chiếm khoảng 1.6% dữ liệu – mức chấp nhận được.

Dựa trên kết quả phân cụm bằng thuật toán DBSCAN và phân tích đặc trưng trung bình của các cụm, ta rút ra được các nhận định như sau: Nhận định: - DBSCAN đã giúp phát hiện **2 cụm nhỏ nhưng có đặc điểm kinh tế vượt trội**, mặc dù tổng thể chỉ có 3 cụm chính. - Việc xác định các nhóm **giàu – trung bình - nghèo** là hợp lý để phục vụ cho các phân tích sâu hơn (ví dụ: gợi ý dịch vụ, đầu tư, hoạch định chính sách).

6.5 Đánh giá

[ ]:
```
mask = labels != -1
X_valid = X_scaled[mask]
labels_valid = labels[mask]

if len(set(labels_valid)) > 1:
    score = silhouette_score(X_valid, labels_valid)
    print(f"Silhouette Score (không tính nhiễu): {score:.4f}")
else:
    print("Không đủ cụm hợp lệ để tính Silhouette Score.")
```

```
Silhouette Score (không tính nhiễu): 0.7164
```

6.6 Kết luận

Mô hình DBSCAN tỏ ra phù hợp với bài toán phân cụm kinh tế vùng khi làm việc với dữ liệu trên. Nhờ khả năng phát hiện hình dạng cụm linh hoạt và loại bỏ nhiễu, DBSCAN giúp phân nhóm rõ ràng theo thu nhập và giá nhà mà không cần xác định trước số cụm. Kết quả cho thấy các cụm được hình thành có ý nghĩa thực tiễn và hỗ trợ tốt cho phân tích chính sách hoặc hệ thống gợi ý theo mức sống.

6.7 Xây dựng demo Recommender System

[14]:
```
# Gán nhãn
dfDBSCAN['cluster'] = dfDBSCAN['cluster'].replace({
    0: 'Nghèo',
    1: 'Trung bình',
    2: 'Giàu',
    -1: 'Nhiễu'
})

# Tạo file csv cho app Recommender System
dfDBSCAN.to_csv("dfDBSCAN_labeled.csv", index=False)
```

Để chạy app, vào terminal: streamlit run app_NguyenThanhCong.py

7 Spectral Clustering(Nguyễn Hoàng Nguyên)

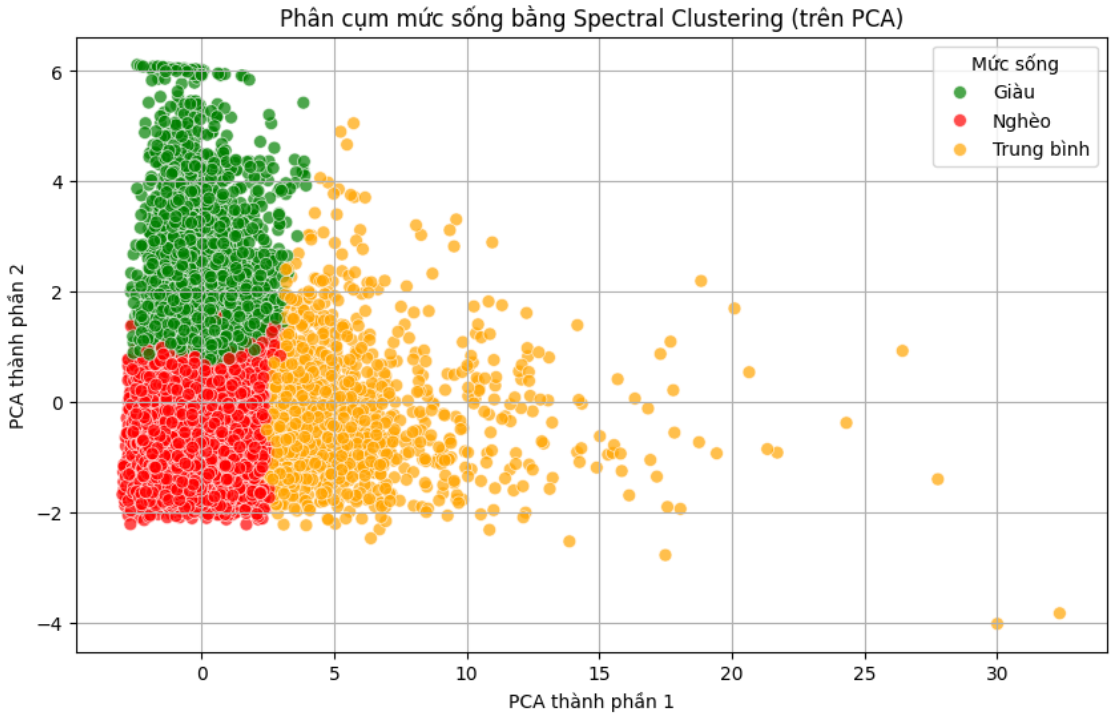
7.1 Lí do lựa chọn

- Mạnh trong các trường hợp dữ liệu không phân tách tuyến tính, cụm có hình dạng phức tạp (phi cầu).
- Không dựa trên giả định về hình dạng cụm như KMeans (cầu, đường kính gần bằng nhau).
- Dùng đồ thị (graph) và ma trận Laplacian để tách các cụm, nên bắt được cấu trúc phi tuyến.

7.2 Huấn luyện và biểu diễn

[14]:
```
# Phân cụm bằng Spectral Clustering
spectral = SpectralClustering(n_clusters=3, affinity='nearest_neighbors',␣
 ↪random_state=42)
labels = spectral.fit_predict(X_scaled)
df['cluster'] = labels

# Giảm chiều dữ liệu bằng PCA để trực quan hóa
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Tính toán mức thu nhập trung bình của mỗi cụm và gán nhãn
```

```
cluster_income = df.groupby('cluster')['median_income'].mean().
  ↪sort_values(ascending=False)
label_map = {
    cluster_income.index[0]: 'Giàu',
    cluster_income.index[1]: 'Trung bình',
    cluster_income.index[2]: 'Nghèo'
}
df['economic_level'] = df['cluster'].map(label_map)

# Biểu đồ PCA
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=X_pca[:, 0],
    y=X_pca[:, 1],
    hue=df['economic_level'],
    palette={'Giàu': 'green', 'Trung bình': 'orange', 'Nghèo': 'red'},
    alpha=0.7,
    s=50
)
plt.title("Phân cụm mức sống bằng Spectral Clustering (trên PCA)")
plt.xlabel("PCA thành phần 1")
plt.ylabel("PCA thành phần 2")
plt.legend(title="Mức sống")
plt.grid(True)
plt.show()
```



[15]:
```
# Hiển thị đặc trưng trung bình cho từng cụm
features = ['median_income', 'median_house_value',
            'housing_median_age', 'total_rooms',
            'total_bedrooms', 'population', 'households']

cluster_summary = df.groupby('economic_level')[features].mean()
print("Đặc trưng trung bình theo mức sống:")
print(cluster_summary)
```

```
Đặc trưng trung bình theo mức sống:
                median_income  median_house_value  housing_median_age  \
economic_level
Giàu                 6.323393       383459.586361           30.608761
Nghèo                3.190167       159341.850073           29.320595
Trung bình           4.122935       208971.313160           15.792400

                total_rooms  total_bedrooms   population   households
economic_level
Giàu            2613.774515      441.847934  1099.974116   420.962668
Nghèo           2109.540016      458.274643  1250.601240   425.311925
Trung bình      8261.155524     1649.256861  4183.483462  1503.426460
```

7.3 Đánh giá và nhận xét

[16]:
```
# Tính Silhouette Score
score = silhouette_score(X_scaled, labels)
print(f"Silhouette Score: {score:.4f}")
```

```
Silhouette Score: 0.3174
```

Với giá trị 0.3174, có thể kết luận rằng mô hình đã tạo ra được sự phân chia cụm tương đối hợp lý. Tuy nhiên, mức phân tách chưa hoàn toàn rõ ràng, vẫn tồn tại một số điểm dữ liệu nằm gần ranh giới giữa các cụm, dẫn đến hiện tượng giao nhau hoặc mờ nhòe giữa các mức sống.

Điều này là hoàn toàn dễ hiểu, vì các yếu tố liên quan đến mức sống như thu nhập, dân cư, số phòng ở, giá nhà,... thường không phân biệt rạch ròi trong thực tế. Chẳng hạn, một khu vực có dân số đông nhưng thu nhập không quá thấp có thể bị "giao thoa" giữa cụm Trung bình và Nghèo. Hoặc những khu vực có giá trị nhà cao do vị trí địa lý nhưng không hẳn phản ánh mức thu nhập thực tế của dân cư.

8 Gaussian Mixture(Nguyễn Trọng Vỹ)

8.1 Lí do lựa chọn

- Là mô hình xác suất, cho phép mỗi điểm dữ liệu thuộc nhiều cụm với xác suất khác nhau (soft clustering).

- Không yêu cầu tất cả các cụm có cùng kích thước hoặc mật độ.
- Có thể ước lượng phương sai, hình dạng elip của cụm (linh hoạt hơn KMeans chỉ tạo cụm cầu).

8.2 Huấn luyện và biểu diễn

[ ]:
```
gmm = GaussianMixture(
    n_components=3,
    covariance_type='spherical',
    n_init=10,
    random_state=42
)
gmm.fit(X_scaled)

# Dự đoán nhãn cụm
df['cluster'] = gmm.predict(X_scaled)
```

[ ]:
```
# Phân tích đặc điểm từng cụm
df.groupby('cluster')[features].mean()
```

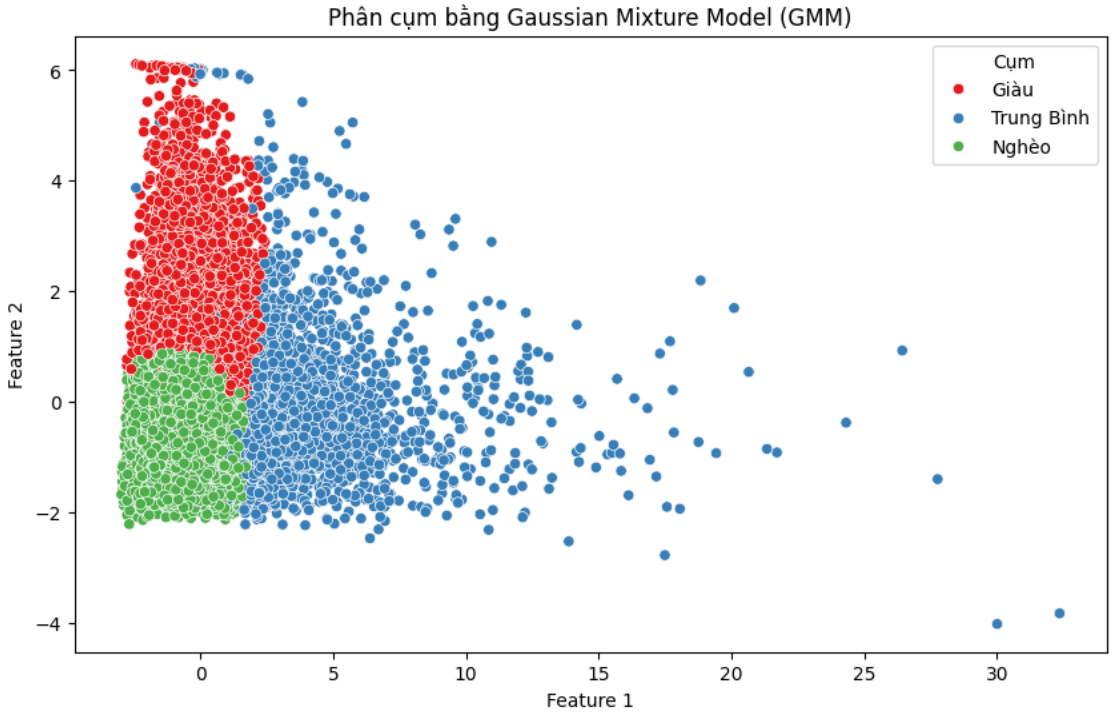
```
         median_income  median_house_value  housing_median_age  total_rooms  \
cluster
0             4.086313       215837.812661           17.860896  6833.696382
1             3.134666       154219.654231           29.979968  1941.155175
2             6.097761       369454.594276           30.123009  2598.341103

         total_bedrooms   population   households
cluster
0           1395.833333  3557.597330  1271.511628
1            421.088329  1160.403614   391.683045
2            449.441265  1123.276944   428.309485
```

Dựa vào phân tích các đặc điểm từng cụm, nhận thấy:

- Cụm 0: Khu vực Trung Bình
- Cụm 1: Khu vực Nghèo
- Cụm 2: Khu vực Giàu

[ ]:
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
label_mapping = {
    0: 'Trung Bình',
    1: 'Nghèo',
    2: 'Giàu'
}
df['cluster_label'] = df['cluster'].map(label_mapping)
```

[ ]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['cluster_label'],␣
↪palette='Set1')
plt.title("Phân cụm bằng Gaussian Mixture Model (GMM)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend(title='Cụm')
plt.show()
```



Dựa vào biểu đồ phân cụm, nhận thấy mô hình vẫn có một số sai sót trong việc phân cụm giữa cụm Trung Bình và Giàu (Thể hiện bằng việc có một số dấu chấm Trung Bình nằm trong cụm Giàu)

8.3 Đánh giá và nhận xét

[21]:
```
score = silhouette_score(X_scaled, df['cluster_label'])
print(f"Silhouette Score: {score:.4f}")
```

```
Silhouette Score: 0.3100
```

Điểm silhouette_score = 0.31, cho thấy mô hình phân cụm ở mức trung bình

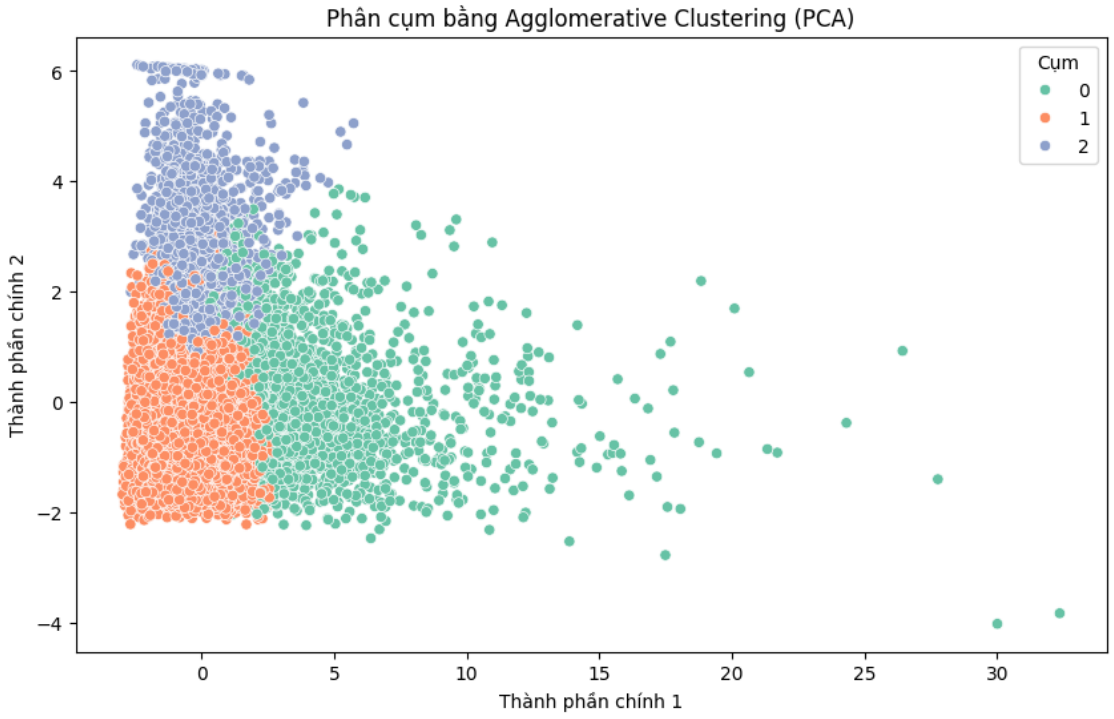
9 Agglomerative Clustering(Vũ Minh Đức)

9.1 Lí do lựa chọn

- Là thuật toán phân cụm phân cấp (hierarchical), không cần xác định số cụm trước khi chạy (có thể cắt cây dendrogram sau).
- Tận dụng tốt khi dữ liệu nhỏ hoặc trung bình, vì trực quan dễ giải thích bằng cây phân cấp.
- Không yêu cầu giả định phân bố cụm, bắt được cụm bất kỳ hình dạng nào.

9.2 Huấn luyện và biểu diễn

[22]:
```
# Khởi tạo mô hình
agglo = AgglomerativeClustering(n_clusters=3)

# Huấn luyện và gán nhãn
df['cluster_agglo'] = agglo.fit_predict(X_scaled)
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=X_pca[:, 0], y=X_pca[:, 1],
    hue=df['cluster_agglo'], palette='Set2'
)
plt.title("Phân cụm bằng Agglomerative Clustering (PCA)")
plt.xlabel("Thành phần chính 1")
plt.ylabel("Thành phần chính 2")
plt.legend(title='Cụm')
plt.show()
```



[23]:
```
df.groupby('cluster_agglo')[features].mean()
```

[23]:

cluster_agglo	median_income	median_house_value	housing_median_age	total_rooms	total_bedrooms	population	households
0	4.208529	238519.698222	19.037001	7107.650168	1436.649688	3611.648727	1311.084094
1	3.350939	178789.458428	30.183960	2047.596797	438.753206	1183.819415	408.590415
2	7.655108	397454.626764	26.051582	2779.068613	413.761557	1122.819951	397.936740

9.3 Đánh giá và nhận xét

[24]:
```
from sklearn.metrics import silhouette_score

score_agglo = silhouette_score(X_scaled, df['cluster_agglo'])
print(f"Silhouette Score (Agglomerative): {score_agglo:.4f}")
```

```
Silhouette Score (Agglomerative): 0.3331
```

Silhouette score ở mức 0.3331, thể hiện cụm khá ổn, tuy chưa phải tối ưu nhất.

Mô hình Agglomerative Clustering hoạt động khá ổn định, đặc biệt khi bạn không muốn giả định trước hình dạng cụm.

Phân cụm tạo ra được 3 vùng địa lý – kinh tế rõ ràng: Nghèo – Trung Bình – Giàu.

10 Kmeans(Nguyễn Ngọc Hải)

10.1 Lý do lựa chọn

- Tốc độ nhanh, hiệu quả tốt với dữ liệu lớn
- Phù hợp khi biết trước số cụm
- Trực quan hóa dễ dàng
- Xử lí dữ liệu trước có thể cải thiện thuật toán

10.2 Chọn tham số và huấn luyện

[ ]:
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
param_grid = {
    'n_clusters': [2, 3, 4, 5, 6],
```

```
    'init': ['k-means++', 'random'],
    'n_init': [10, 20, 30]
}

# 6. GridSearchCV
grid_search = GridSearchCV(
    param_grid=param_grid,
    scoring=None,
    cv=[(slice(None), slice(None))]
)

grid_search.fit(X_pca)
kmeans = KMeans(n_clusters=best_params['n_clusters'],
    init=best_params['init'],
    n_init=best_params['n_init'],
    n_init='auto')
df['cluster'] = kmeans.fit_predict(X_scaled)
```

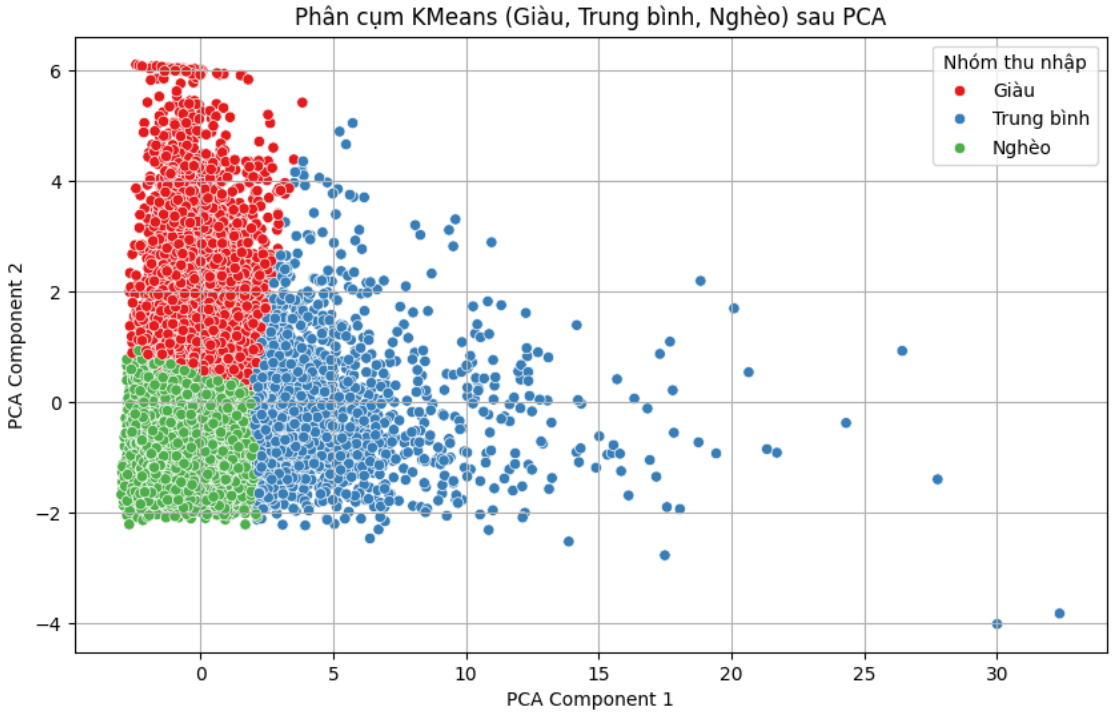
10.3 Phân tích và biểu diễn

[ ]:
```
df.groupby('cluster')[features].mean()
```

[ ]:
```
         median_income  median_house_value  housing_median_age  total_rooms  \
cluster
0             5.989685       358119.731379           30.280111  2607.065174
1             4.035541       212631.482613           16.977871  7387.296628
2             3.055094       149440.023430           29.656855  1978.952843

         total_bedrooms   population   households
cluster
0            452.820325  1140.173138   431.246434
1           1500.533720  3809.563751  1370.374605
2            434.229925  1195.951286   402.389041
```

[36]:
```
label_mapping = {
    0: 'Giàu',
    1: 'Trung bình',
    2: 'Nghèo'
}
df['cluster_label'] = df['cluster'].map(label_mapping)
# 7. Trực quan hóa kết quả phân cụm trên không gian PCA
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['cluster_label'],␣
 ↪palette='Set1')
plt.title('Phân cụm KMeans (Giàu, Trung bình, Nghèo) sau PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
```

```
plt.legend(title='Nhóm thu nhập')
plt.grid(True)
plt.show()
```



Cụm "Giàu" (đỏ) và "Nghèo" (xanh lá) có ranh giới khá rõ rệt theo trục PCA Component 2.

Cụm "Trung bình" (xanh dương) trải rộng và xen lẫn một phần với hai cụm còn lại, cho thấy biên giới giữa trung bình và hai nhóm cực trị không hoàn toàn tách biệt.

10.4 Đánh giá

[30]:
```
labels = kmeans.labels_

# Tính Silhouette Score
score = silhouette_score(X, labels)
print(f"Silhouette Score: {score:.4f}")
```

```
Silhouette Score: 0.2362
```

Với Silhouette Score = 0.2362, chất lượng phân cụm hiện tại ở mức trung bình thấp, cho thấy biên giới giữa các cụm còn mờ nhạt.

10.5 Kết luận

- Để nâng cao hiệu quả phân cụm, có thể cân nhắc điều chỉnh số lượng cụm

- Lựa chọn lại đặc trưng đầu vào hoặc thử nghiệm các thuật toán phân cụm khác như DBSCAN, Gaussian Mixture hoặc Agglomerative Clustering.

11 Kết luận

11.1 Tổng kết

Thuật toán	Silhouette Score	Nhận xét
DBSCAN	7.164	Cao nhất trong các thuật toán, cho thấy khả năng phân cụm rất tốt, các cụm tách biệt rõ và dữ liệu trong mỗi cụm khá đồng nhất.
Gaussian Mixture	3.100	Hiệu quả ở mức trung bình, cụm chưa thật sự tách biệt rõ ràng.
Spectral Clustering	3.174	Gần tương tự Gaussian Mixture, khả năng tách cụm ở mức trung bình.
Agglomerative Clustering	3.331	Tốt hơn một chút so với Gaussian Mixture và Spectral Clustering nhưng vẫn chưa đạt mức tốt.
KMeans	2.362	Thấp nhất, cho thấy cụm chưa rõ ràng, khoảng cách giữa các cụm không lớn, khả năng phân tách kém.

Nhận xét:

- DBSCAN là lựa chọn tốt nhất cho bộ dữ liệu này khi xét về Silhouette Score.
- Các thuật toán khác (Gaussian Mixture, Spectral, Agglomerative, KMeans) cho kết quả khá thấp → có thể không phù hợp với phân bố dữ liệu hiện tại hoặc cần tinh chỉnh siêu tham số.
- KMeans cho kết quả kém nhất, cho thấy dữ liệu có thể không tuân theo dạng hình cầu (sphere) mà KMeans giả định.

11.2 Giải pháp

- Tiền xử lý dữ liệu.
- Điều chỉnh siêu tham số.
- Kết hợp nhiều phương pháp.
- Sử dụng Clustering Ensemble (kết hợp nhiều thuật toán).
- Có thể chia ra các cụm khác.

11.3 Hướng phát triển

- Đánh giá đa tiêu chí: Kết hợp nhiều chỉ số đánh giá như Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Score.
- Mở rộng và kiểm tra tính ổn định: Áp dụng trên tập dữ liệu lớn hơn hoặc dữ liệu thời gian thực.

- Tự động chọn thuật toán: Xây dựng pipeline tự động thử nhiều thuật toán và chọn ra mô hình tốt nhất.
- Nghiên cứu Deep Clustering: Sử dụng autoencoder kết hợp clustering (Deep Embedded Clustering - DEC) để xử lý dữ liệu phức tạp.

11.4 Bài học

- Kiến thức cơ bản về Python và cách sử dụng các thư viện như Pandas, NumPy, Matplotlib, Scikit-learn.
- Cách xử lý dữ liệu đầu vào, bao gồm làm sạch, chuẩn hóa và chọn lọc đặc trưng để phục vụ cho việc phân cụm.
- Nguyên lý hoạt động của một số thuật toán phân cụm như KMeans, DBSCAN, Agglomerative, Gaussian Mixture.
- Cách áp dụng các chỉ số đánh giá như Silhouette Score, Davies-Bouldin Index để đo lường chất lượng phân cụm.
- Quy trình triển khai và đánh giá một bài toán clustering hoàn chỉnh.