THỰC HÀNH: CÁC GIẢI THUẬT PHÂN CỤM CƠ BẢN

A. MỤC TIÊU

Hướng dẫn thực hiện phân cụm dữ liệu với giải thuật K-means

Hướng dẫn thực hiện phân cụm đa cấp

Hướng dẫn ôn tập các kiến thức lý thuyết về K-means và phân cụm đa cấp

B. KẾT CẤU

Nội dung bao gồm 2 phần là

- Giải thuật K-means
- Giải thuật phân cụm đa cấp

C. NỘI DUNG

3.1. GIẢI THUẬT K-MEANS

3.1.1. Ôn tập lý thuyết

+ Giải thuật K-Means hoạt động như thế nào? Hãy giải thích các bước chính trong quy trình phân cụm

+ Tại sao cần chọn số lượng cụm (K) trước khi chạy K-Means? Làm thế nào để xác định giá trị K tối ưu?

+ Hàm mục tiêu (objective function) của K-Means là gì? Nó đo lường điều gì trong quá trình phân cụm?

+ Những hạn chế của K-Means là gì? Trong trường hợp nào K-Means có thể cho kết quả không tốt?

+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai K-Means Clustering không? Hãy mô tả các bước thực hiện

+ Sử dụng phương pháp nào trong Python để chọn số cụm K tối ưu (ví dụ: Elbow Method, Silhouette Score)? Hãy chia sẻ một đoạn code mẫu

+ K-Means nhạy cảm với giá trị khởi tạo (initial centroids), bạn sẽ làm gì để đảm bảo kết quả ổn định (ví dụ: K-Means++)?

+ Làm thế nào để đánh giá chất lượng của các cụm được tạo bởi K-Means? Bạn sử dụng chỉ số nào (ví dụ: Silhouette Score, Within-Cluster Sum of Squares)?

3.1.2. Bài tập mẫu

Bài toán 1: Thực hiện các nhiệm vụ trong bài toán 1 để làm quen với việc xây dựng mô hình dựa vào thuật toán phân cụm K-Means trên tập dữ liệu Mall-Customer. Dữ liệu lấy tại https://www.kaggle.com/code/fadymamdouh01/mall-customer-segmentation-data-clustering.

Nhiệm vụ 1: Thực hiện giải thuật K-Means để phân cụm dữ liệu khách.

1. Import các gói thư viện và nạp dữ liệu vào notebook

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
warnings.filterwarnings('ignore')
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

df = pd.read_csv('Mall_Customers.csv')
# Transform Gender to 0's and 1's
df['Gender'] = df['Gender'].map({'Male': 1, "Female": 0})
df.info()
# Data for cluster
X = df.iloc[:, -4:]
```

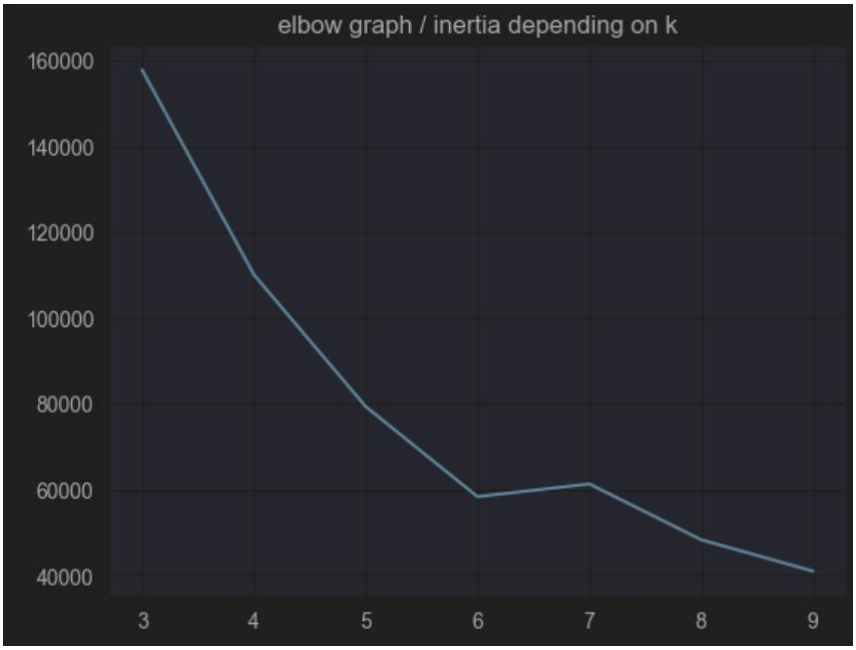
2. Xây dựng mô hình K-Means

```
# We will create K-Means Models `iteratively between k values of 3 to 10` and at each step, capture the `Silhoutte Score` and `Inertia (Sum of Squared Distances)`
km_inertias, km_scores = [], []
for k in range(3, 10):
    km = KMeans(n_clusters=k).fit(X)
    km_inertias.append(km.inertia_)
    km_scores.append(silhouette_score(X, km.labels_))
    print(f"Processing K-Means with k = {k}, Intertia = {km.inertia_},
            Silhoutte Score = {silhouette_score(X, km.labels_)}")
km_inertias
```

3. Tìm số cụm tối ưu cho mô hình K-Mean bằng phương pháp Elbow

```
# sns.lineplot(range(3, 10), km_inertias) PHUOCNT
sns.lineplot(x=range(3, 10), y=km_inertias)
plt.title('elbow graph / inertia depending on k')
plt.show()
```

Kết quả thực hiện và ta chọn số cụm là k = 6

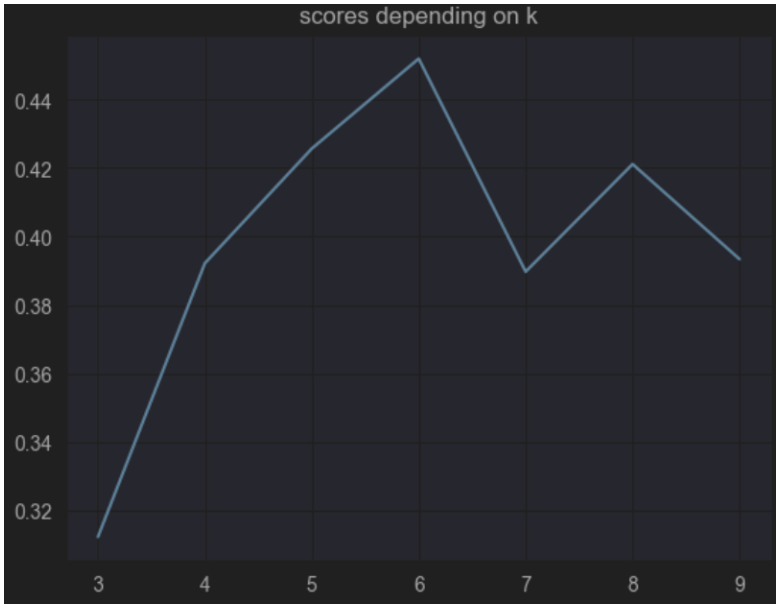


Hình 3.1 - Biểu đồ Elbow

4. Tìm số cụm tối ưu bằng phương pháp Silhoutte

```
sns.lineplot(x=range(3, 10), y=km_scores)
plt.title('scores depending on k')
plt.show()
```

Kết quả thực hiện và chọn số cụm k = 6



Hình 3.2 - Biểu đồ Silhoutte

4. Gán nhãn cho các mẫu dữ liệu dựa vào mô hình K-Means với số cụm là 6

```
km = KMeans(n_clusters=6).fit(X)
```

```
#Assign the Cluster Labels to the Data
X['Label'] = km.labels_
#Info for each cluster
for k in range(6):
    print(f'Cluster nb : {k}')
    print(X[X.Label == k].describe().iloc[:, 1:-1])
    print('\n\n')
```

Bài toán 2: Thực hiện phân cụm dữ liệu trên tập dữ liệu iris-data. Dữ liệu lấy từ lấy từ https://www.kaggle.com/code/xvivancos/tutorial-knn-in-the-iris-data-set

Nhiệm vụ 1: Phân cụm dữ liệu trên tập dữ liệu iris-data.

1. Import thư viện và load dữ liệu

```
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns
import plotly.graph_objects as go
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import Kmeans

iris = pd.read_csv("Iris.csv")
iris.drop('Id',inplace=True,axis=1)

X = iris.iloc[:,:-1].values #Set our training data
y = iris.iloc[:,-1].values #We'll use this just for visualization as
clustering doesn't require labels

iris.head().style.background_gradient(cmap
                                =sns.cubehelix_palette(as_cmap=True))
```

2. EDA và xử lý dữ liệu

```
# Data Distribution => The Data is perfectly balanced
fig = px.pie(iris,
'Species',color_discrete_sequence=['#491D8B','#7D3AC1','#EB548C'],title=
'Data Distribution',template='plotly')
fig.show()


# Phân tích biến Sepal-Length với box-plot
fig = px.box(data_frame=iris, x='Species',y='SepalLengthCm',
        color='Species',
```

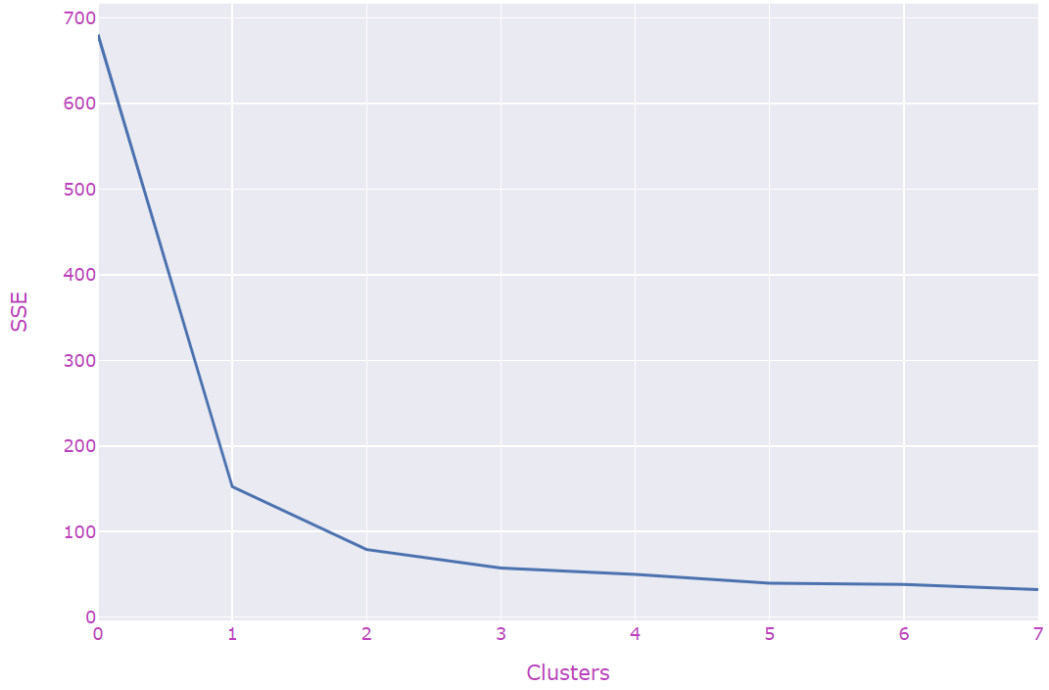
```
        color_discrete_sequence=['#29066B','#7D3AC1','#EB548C'],
        orientation='v')
fig.show()

# Phân tích biến SepalLengthCm với histogram
fig = px.histogram(data_frame=iris, x='SepalLengthCm',
           color='Species',
           color_discrete_sequence=['#491D8B','#7D3AC1','#EB548C'],
           nbins=50)
fig.show()

# Phân tích biến SepalWidth với box plot
fig = px.box(data_frame=iris, x='Species',
         y='SepalWidthCm', color='Species',
         color_discrete_sequence=['#29066B','#7D3AC1','#EB548C'],
         orientation='v')
fig.show()

# Phân tích 2 biến SepalLengthCm và SepalWidthCm
fig = px.scatter(data_frame=iris, x='SepalLengthCm',y='SepalWidthCm'
     ,color='Species',size='PetalLengthCm',
     template='seaborn',
     color_discrete_sequence=['#491D8B','#7D3AC1','#EB548C'],)
fig.update_layout(width=800, height=600,
                  xaxis=dict(color="#BF40BF"),
                  yaxis=dict(color="#BF40BF"))
fig.show()

# Phân tích 2 biến PetalLengthCm và PetalWidthCm
fig = px.scatter(data_frame=iris, x='PetalLengthCm',y='PetalWidthCm'
           ,color='Species',size='SepalLengthCm',
           template='seaborn',
           color_discrete_sequence=['#491D8B','#7D3AC1','#EB548C'],)
fig.update_layout(width=800, height=600,
                  xaxis=dict(color="#BF40BF"),
                 yaxis=dict(color="#BF40BF"))
fig.show()
```

3. Tìm số cluster tối ưu cho mô hình K-Means

```
sse = []
```

```
for i in range(1,9):
    kmeans = KMeans(n_clusters=i , max_iter=300)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)
fig = px.line(y=sse,template="seaborn",title='Eblow Method')
fig.update_layout(width=800, height=600,
title_font_color="#BF40BF",
xaxis=dict(color="#BF40BF",title="Clusters"),
yaxis=dict(color="#BF40BF",title="SSE"))
# Cluster = 3 là số cluster tối ưu cho dữ liệu Iris
```

Kết quả thực hiện



4. Xây dựng mô hình K-Means với số cluster tối ưu = 3

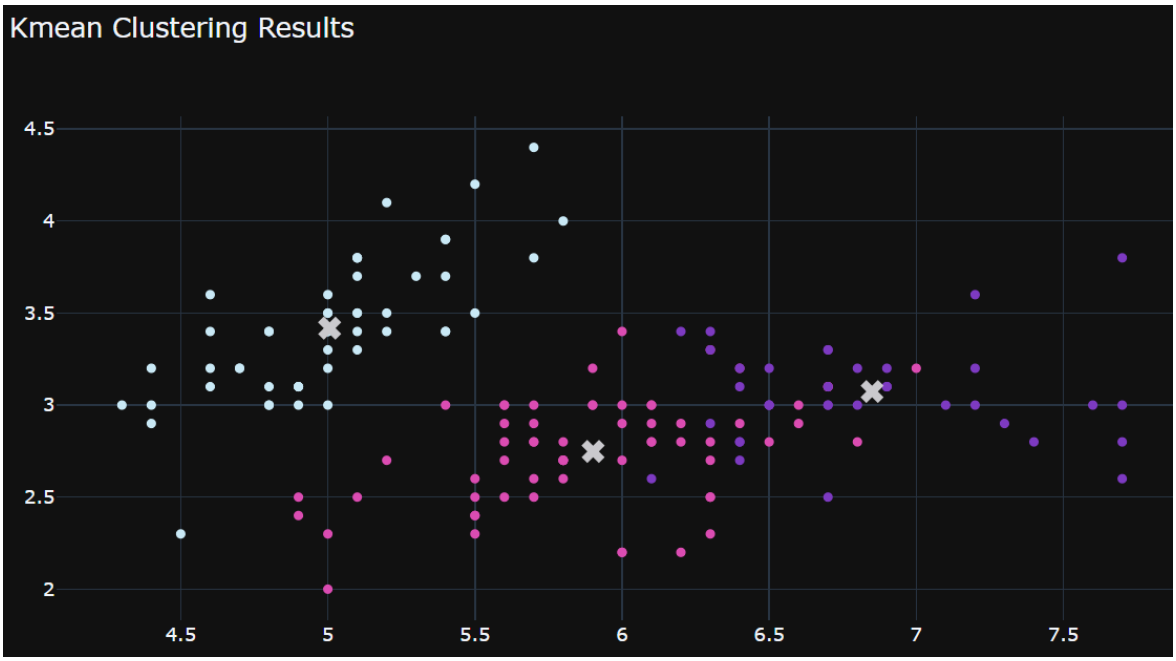
```
kmeans = KMeans(n_clusters = 3,
        init = 'k-means++',
        max_iter = 300, n_init = 10, random_state = 0)
clusters = kmeans.fit_predict(X)
```

5. Trình bày kết quả trực quan các cluster

```
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=X[clusters == 0, 0], y=X[clusters == 0, 1],
    mode='markers',marker_color='#DB4CB2',name='Iris-setosa'
))
```

```
fig.add_trace(go.Scatter(
    x=X[clusters == 1, 0], y=X[clusters == 1, 1],
    mode='markers',marker_color='#c9e9f6',name='Iris-versicolour'
))
fig.add_trace(go.Scatter(
    x=X[clusters == 2, 0], y=X[clusters == 2, 1],
    mode='markers',marker_color='#7D3AC1',name='Iris-virginica'
))
fig.add_trace(go.Scatter(
    x=kmeans.cluster_centers_[:, 0], y= kmeans.cluster_centers_[:,1],
    mode='markers',marker_color='#CAC9CD',marker_symbol=4,
marker_size=13,name='Centroids'
))
fig.update_layout(template='plotly_dark',width=1000,
height=500,title='Kmean Clustering Results')
```

Kết quả đạt được



3.1.3. Bài tập thực hành 1

Xây dựng mô hình phân cụm K-means trên tập dữ liệu chim cánh cụt. Dữ liệu lấy tại https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering

3.1.4. Bài tập thực hành 2

Xây dựng mô hình phân cụm K-means trên tập dữ liệu mua sắm tại siêu thị. Dữ liệu lấy tại https://www.kaggle.com/datasets/hellbuoy/online-retail-customer-clustering

3.2. GIẢI THUẬT PHÂN CỤM ĐA CẤP

3.2.1. Ôn tập lý thuyết

+ Giải thuật phân cụm đa cấp hoạt động như thế nào? Hãy giải thích sự khác biệt giữa phân cụm đa cấp hợp nhất (agglomerative) và phân tách (divisive)

+ Các phương pháp liên kết (linkage) như single linkage, complete linkage, average linkage, và Ward's method khác nhau ra sao? Khi nào nên sử dụng từng loại?

+ Dendrogram trong phân cụm đa cấp là gì? Làm thế nào để sử dụng nó để chọn số lượng cụm?

+ Phân cụm đa cấp có thể áp dụng cho dữ liệu phi số (non-numeric data) như thế nào? Hãy giải thích

+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để triển khai phân cụm đa cấp hợp nhất (agglomerative clustering) không? Hãy mô tả các bước thực hiện

+ Làm thế nào để vẽ dendrogram trong Python sử dụng thư viện như scipy hoặc matplotlib? Hãy chia sẻ một đoạn code mẫu

+ Các lớp trong gói Scipy hỗ trợ phân cụm đa cấp? và so sánh giữa cách tiếp cận scikit-learn và cách tiếp cận sử dụng Scipy

3.2.2. Bài làm mẫu

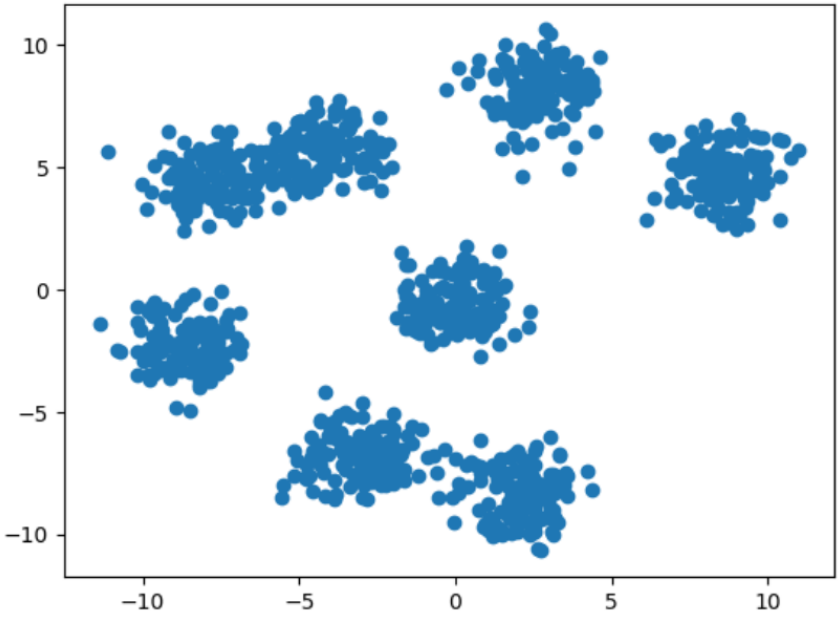
Bài toán 1: Tìm hiểu một số khái niệm cơ bản của phân cụm đa cấp

Nhiệm vụ 1: Tìm hiểu một số khái niệm cơ bản của phân cụm đa cấp

1. Import thư viện, tạo dữ liệu giả (dummy data) và hiển thị dữ liệu

```
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
%matplotlib inline
# Generate a random cluster dataset to experiment on. X = coordinate points, y = cluster labels (not needed)
X, y = make_blobs(n_samples=1000, centers=8, n_features=2, random_state=800)
plt.scatter(X[:,0], X[:,1])
plt.show()
```

Kết quả thực hiện



Hình 3.3 - Biểu đồ 2D hiển thị các cụm dữ liệu

2. Hiển thị ma trận khoảng cách giữa các điểm dữ liệu với điểm trung tâm

```
# Generate distance matrix with 'linkage' function
distances = linkage(X, method="centroid", metric="euclidean")
print(distances)
```

Kết quả thực hiện là một ma trận khoảng cách như bên dưới

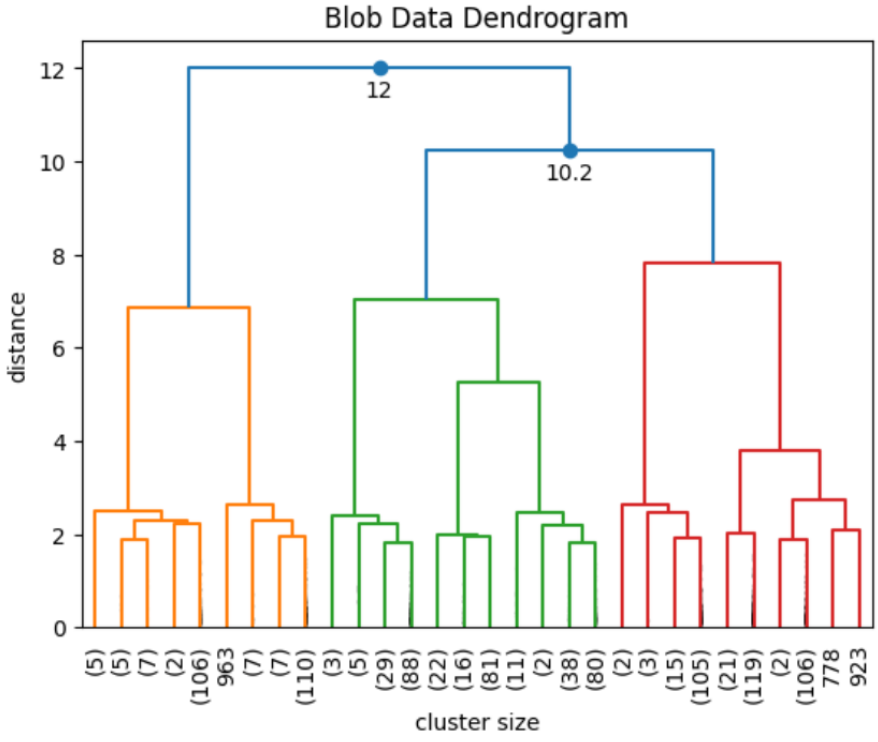
```
[[5.720e+02 7.620e+02 7.694e-03 2.000e+00]
 [3.000e+01 1.960e+02 8.879e-03 2.000e+00]
 [5.910e+02 8.700e+02 1.075e-02 2.000e+00]
 ...
 [1.989e+03 1.992e+03 7.812e+00 3.750e+02]
 [1.995e+03 1.996e+03 1.024e+01 7.500e+02]
 [1.994e+03 1.997e+03 1.200e+01 1.000e+03]]
```

3. Vẽ hình dendrogram từ ma trận khoảng cách

```
# Take normal dendrogram output and stylize in cleaner way
def annotated_dendrogram(*args, **kwargs):
    # Standard dendrogram from SciPy
    scipy_dendro = dendrogram(*args, truncate_mode='lastp',
                      show_contracted=True,leaf_rotation=90.)
    plt.title('Blob Data Dendrogram')
    plt.xlabel('cluster size')
```

```
    plt.ylabel('distance')
    for i, d, c in zip(scipy_dendro['icoord'], scipy_dendro['dcoord'],
                       scipy_dendro['color_list']):
        x = 0.5 * sum(i[1:3])
        y = d[1]
        if y > 10:
            plt.plot(x, y, 'o', c=c)
            plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),
                         textcoords='offset points',
                         va='top', ha='center')
    return scipy_dendro
dn = annotated_dendrogram(distances)
plt.show()
```

Kết quả thực hiện

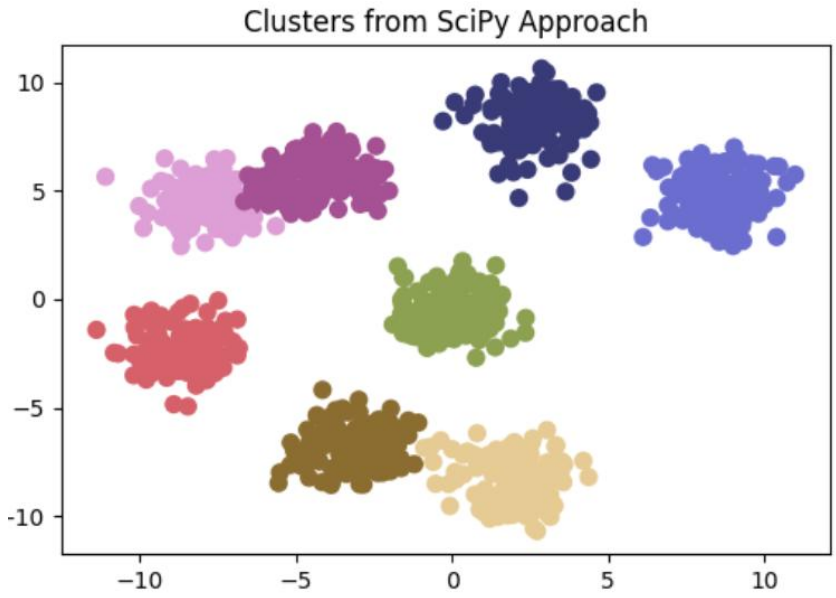


Hình 3.4 - Biểu đồ Dendrogram

4. Vẽ các cụm dữ liệu dựa ma trận khoảng cách và thông tin từ dendrogram với hàm fcluster

```
scipy_clusters = fcluster(distances, 3, criterion="distance")
plt.figure(figsize=(6,4))
plt.title("Clusters from SciPy Approach")
plt.scatter(X[:, 0], X[:, 1], c = scipy_clusters ,s=50, cmap='tab20b')
plt.show()
```

Kết quả thực hiện



Hình 3.5 - Biểu đồ các cụm dữ liệu với cách tiếp cận SciPy

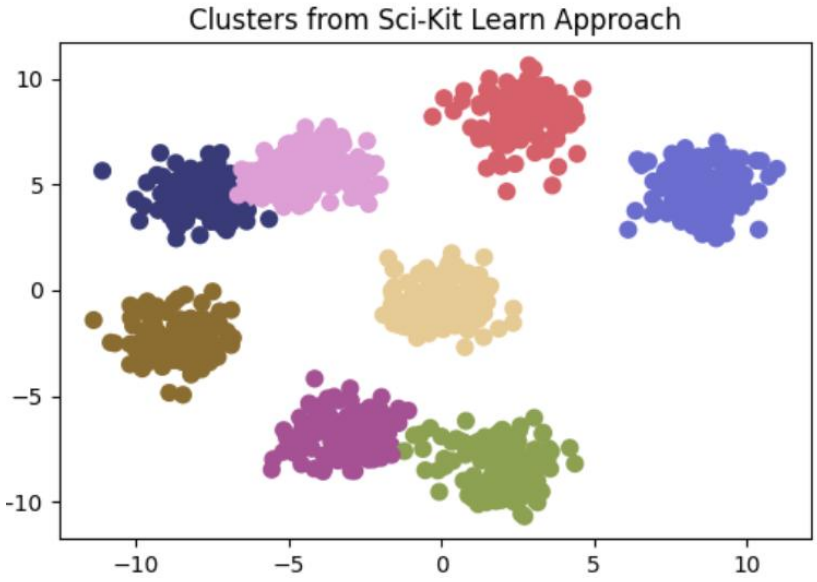
5. Vẽ các cụm dữ liệu với các phương pháp linkage khác nhau

```
methods = ['centroid', 'single', 'complete', 'average', 'weighted']
for method in methods:
    distances = linkage(X, method=method, metric="euclidean")
    clusters = fcluster(distances, 3, criterion="distance")
    plt.title('linkage: ' + method)
    plt.scatter(X[:,0], X[:,1], c=clusters, cmap='tab20b')
    plt.show()
```

6. Xây dựng mô hình phân cụm đa cấp theo tiếp cận của thư viện scikit-learn

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

ac = AgglomerativeClustering(n_clusters = 8, linkage="average")
#distances = linkage(X, method="centroid", metric="euclidean")
sklearn_clusters = ac.fit_predict(X)
scipy_clusters = fcluster(distances, 3, criterion="distance")
plt.figure(figsize=(6,4))
plt.title("Clusters from Sci-Kit Learn Approach")
plt.scatter(X[:, 0], X[:, 1], c = sklearn_clusters ,s=50, cmap='tab20b')
plt.show()
```

Kết quả thực hiện



Hình 3.6 - Biểu đồ các cụm dữ liệu với cách tiếp cận scikit-learn

Bài toán 2: Phân cụm đa cấp trên tập dữ liệu rượu vang. Dữ liệu lấy từ https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering/data

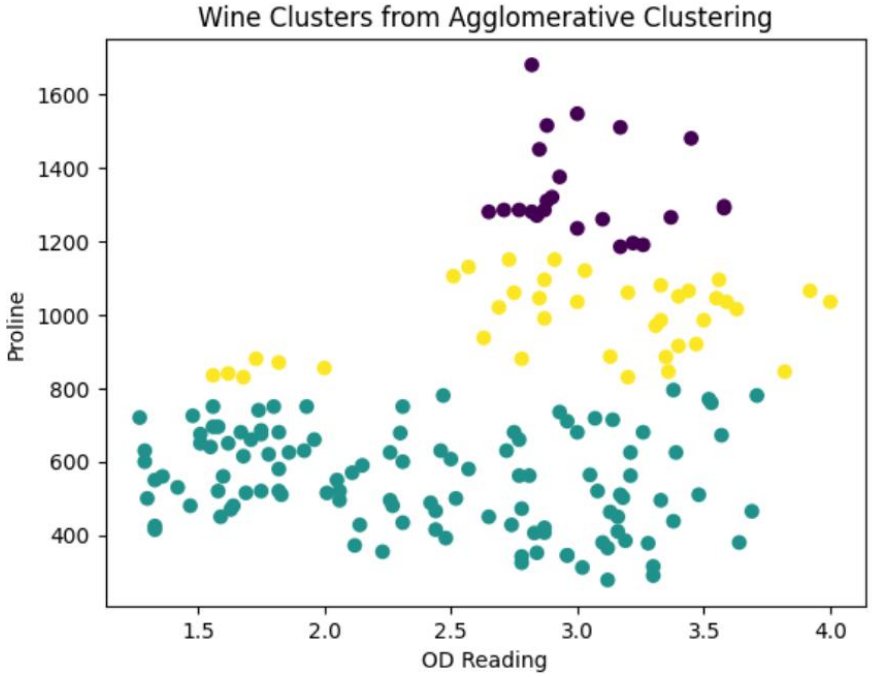
1. Import & tải dữ liệu rượu vang lên notebook

```
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
import pandas as pd
import matplotlib.pyplot as plt
wine_df = pd.read_csv("wine_data.csv")
```

2. Xây dựng mô hình phân cụm đa cấp theo cách tiếp cận scikit-learn và hiển thị kết quả phân cụm trên dữ liệu

```
ac = AgglomerativeClustering(3, linkage='average')
ac_clusters = ac.fit_predict(wine_df)
# Hiển thị các cụm sau khi phân cấp theo cách tiếp cận của scikit-learn
plt.scatter(wine_df.values[:,0], wine_df.values[:,1], c=ac_clusters)
plt.title("Wine Clusters from Agglomerative Clustering")
plt.xlabel("OD Reading")
plt.ylabel("Proline")
plt.show()
```

Kết quả thực hiện



Hình 3.7 – Biểu đồ thông tin các cụm sau khi tiến hành phân cụm đa cấp với scikit-learn

3. Phân cụm sử dụng K-Means và so sánh với kết quả phân cụm đa cấp

```
km = KMeans(3)
km_clusters = km.fit_predict(wine_df)
# Hiển thị kết quả sau khi phân cụm K-mean
plt.scatter(wine_df.values[:,0], wine_df.values[:,1], c=km_clusters)
plt.title("Wine Clusters from K-Means Clustering")
plt.xlabel("OD Reading")
plt.ylabel("Proline")
plt.show()
```

Kết quả thực hiện:

Hình 3.8 - Biểu đồ các cụm dữ liệu sau khi phân cụm với K-Means (3 cụm)

4. So sánh kết quả phân cụm của 2 mô hình với phương pháp tính Silhouette

```
# Calculate Silhouette Scores
print("Silhouette Scores for Wine Dataset:\n")
print("K-Means Clustering: ", silhouette_score(wine_df, km_clusters))
print("Agg Clustering: ", silhouette_score(wine_df, ac_clusters))
```

Kết quả thực hiện

```
Silhouette Scores for Wine Dataset:

K-Means Clustering:  0.5809421087616886
Agg Clustering:  0.5988495817462
```

3.2.3. Bài tập thực hành 1

Xây dựng mô hình phân cụm đa cấp trên tập dữ liệu chim cánh cụt. Dữ liệu lấy tại https://www.kaggle.com/code/youssefaboelwafa/clustering-penguins-species-k-means-clustering

3.2.4. Bài tập thực hành 2

Xây dựng mô hình phân cụm đa cấp trên tập dữ liệu mua sắm tại siêu thị. Dữ liệu lấy tại https://www.kaggle.com/datasets/hellbuoy/online-retail-customer-clustering

D. TÓM TẮT

Với 2 phần là phân cụm với K-means và phân cụm đa cấp được trình bày trong chương đã phần nào giúp sinh viên có thể tiến hình triển khai bài toán phân cụm trên một số tập dữ liệu cơ bản.

Với việc thực hành giải quyết các bài tập trên lớp và các bài tập ở nhà giúp sinh viên hiểu rõ hơn các khái niệm lý thuyết đã học và áp dụng tốt vào việc giải quyết các **bài toán phân cụm** trong thực tế.