ML-Assignment-5-Clustering Algorithm

1.

File Edit View Run Kernel Settings Help

Code

[1]:
```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np

# Load the Iris dataset
iris = load_iris()

# Create a DataFrame from the dataset
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Display the first few rows of the dataset
df.head()
```

[1]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

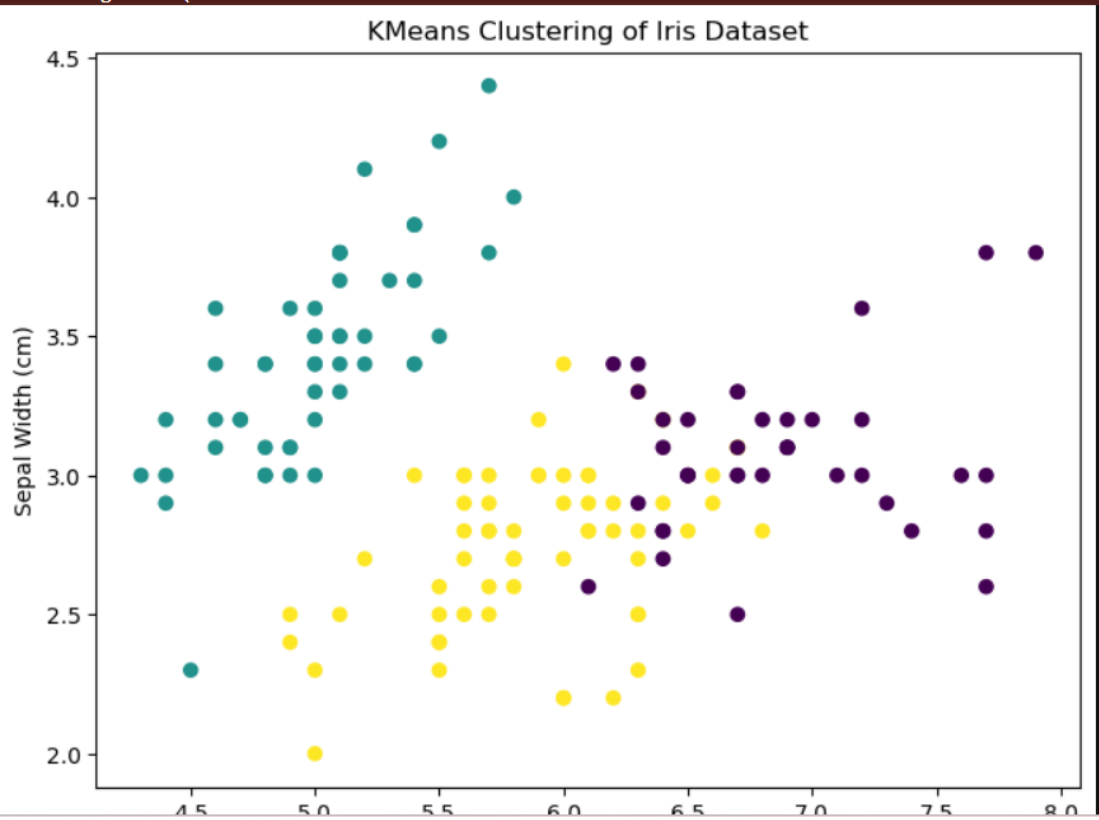
2.

```
a. Brief Description of KMeans Clustering
KMeans clustering is an iterative algorithm that partitions data into k clusters. Each point is assigned to the nea
the centroid is updated based on the points assigned to it. This process repeats until convergence (when centroids

b. Why KMeans Might be Suitable for the Iris Dataset
The Iris dataset has multiple distinct classes, and KMeans can effectively separate them into clusters based on fea
Since the number of clusters is known (3 species of Iris), KMeans is a suitable method to partition the dataset.

c. Apply KMeans Clustering
```

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Applying KMeans with 3 clusters (since there are 3 species in the Iris dataset)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

# Assigning cluster labels to the data
df['Cluster'] = kmeans.labels_

# Visualizing the clusters (using two features: sepal length and sepal width)
plt.figure(figsize=(8, 6))
plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c=df['Cluster'], cmap='viridis', marker='o')
plt.title('KMeans Clustering of Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

3.

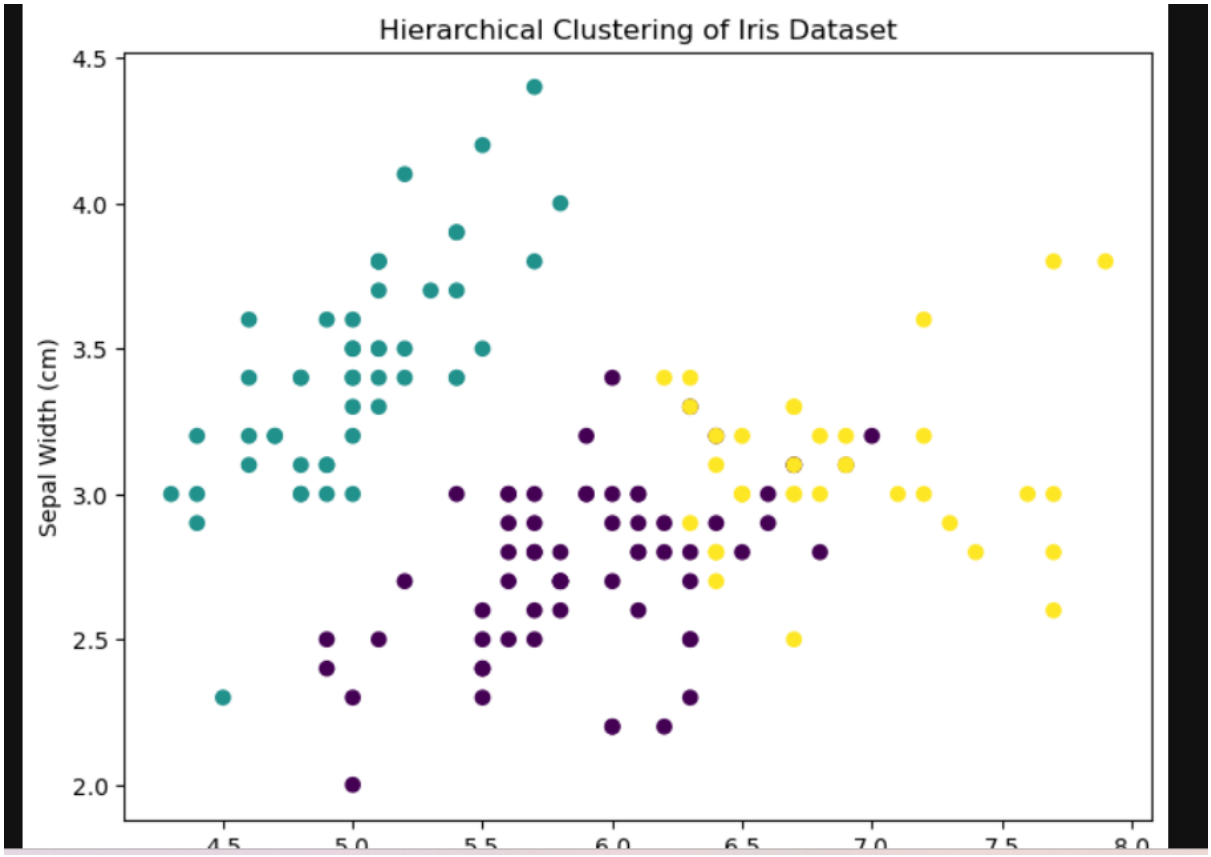
```
C:\Users\ARCHANA\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is kno
n there are less chunks than available threads. You can avoid it by setting the environment variable OMP
  warnings.warn(
```



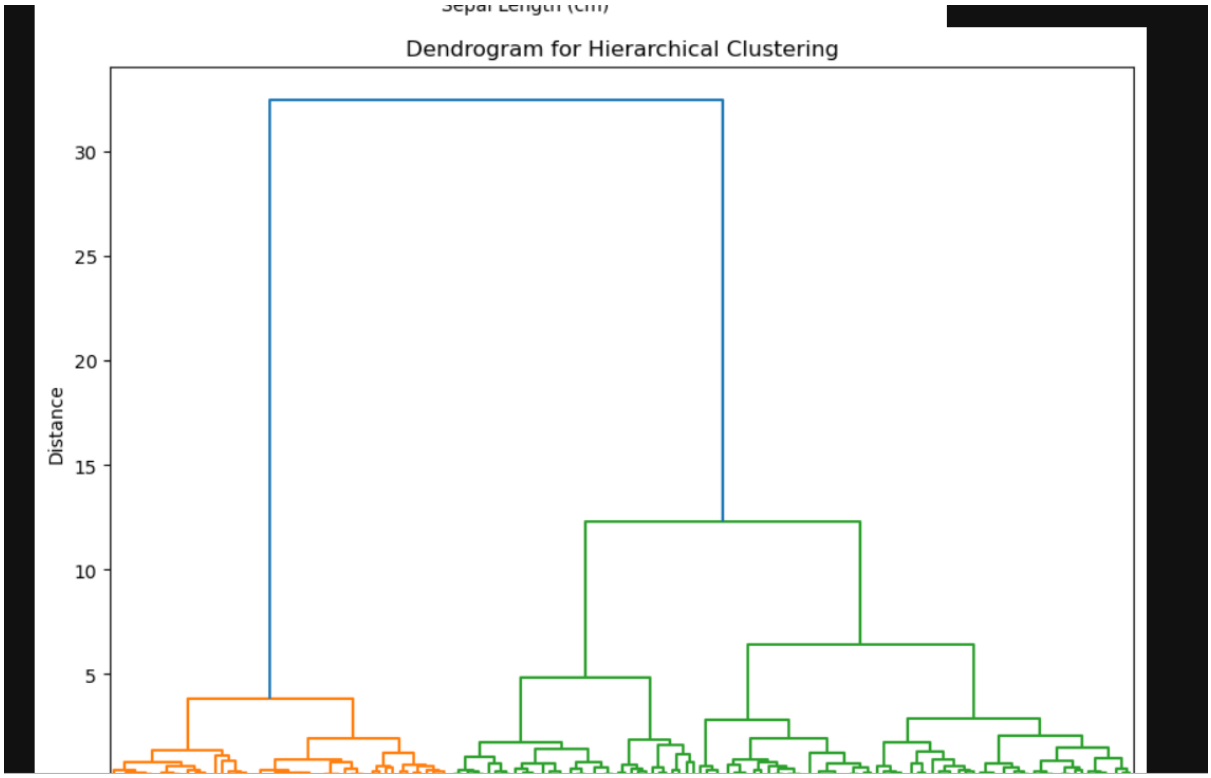
4.

```
[7]: from sklearn.cluster import AgglomerativeClustering
     from scipy.cluster.hierarchy import dendrogram, linkage

     # Perform hierarchical clustering
     hierarchical = AgglomerativeClustering(n_clusters=3)
     df['Hierarchical_Cluster'] = hierarchical.fit_predict(X)

     # Visualizing the clusters (using sepal length and sepal width)
     plt.figure(figsize=(8, 6))
     plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c=df['Hierarchical_Cluster'], cmap='viridis', marker='o
     plt.title('Hierarchical Clustering of Iris Dataset')
     plt.xlabel('Sepal Length (cm)')
     plt.ylabel('Sepal Width (cm)')
     plt.show()

     # Plotting dendrogram for Hierarchical clustering
     linked = linkage(X, 'ward')
     plt.figure(figsize=(10, 7))
     dendrogram(linked)
     plt.title('Dendrogram for Hierarchical Clustering')
     plt.xlabel('Index')
     plt.ylabel('Distance')
     plt.show()
```

5.



6.



Final Explanation

- The Iris dataset contains four features: sepal length, sepal width, petal length, and petal width. These can be used to group the flowers into clusters based on similarity.
- KMeans and Hierarchical clustering methods are used to cluster the data into 3 groups (since we know there are 3 species in the Iris dataset).
- KMeans gives a fixed number of clusters, while hierarchical clustering provides flexibility and a dendrogram for better understanding of cluster formation.

This approach covers all the requirements, including clustering implementation with KMeans and Hierarchical clustering, as well as their respective visualizations.