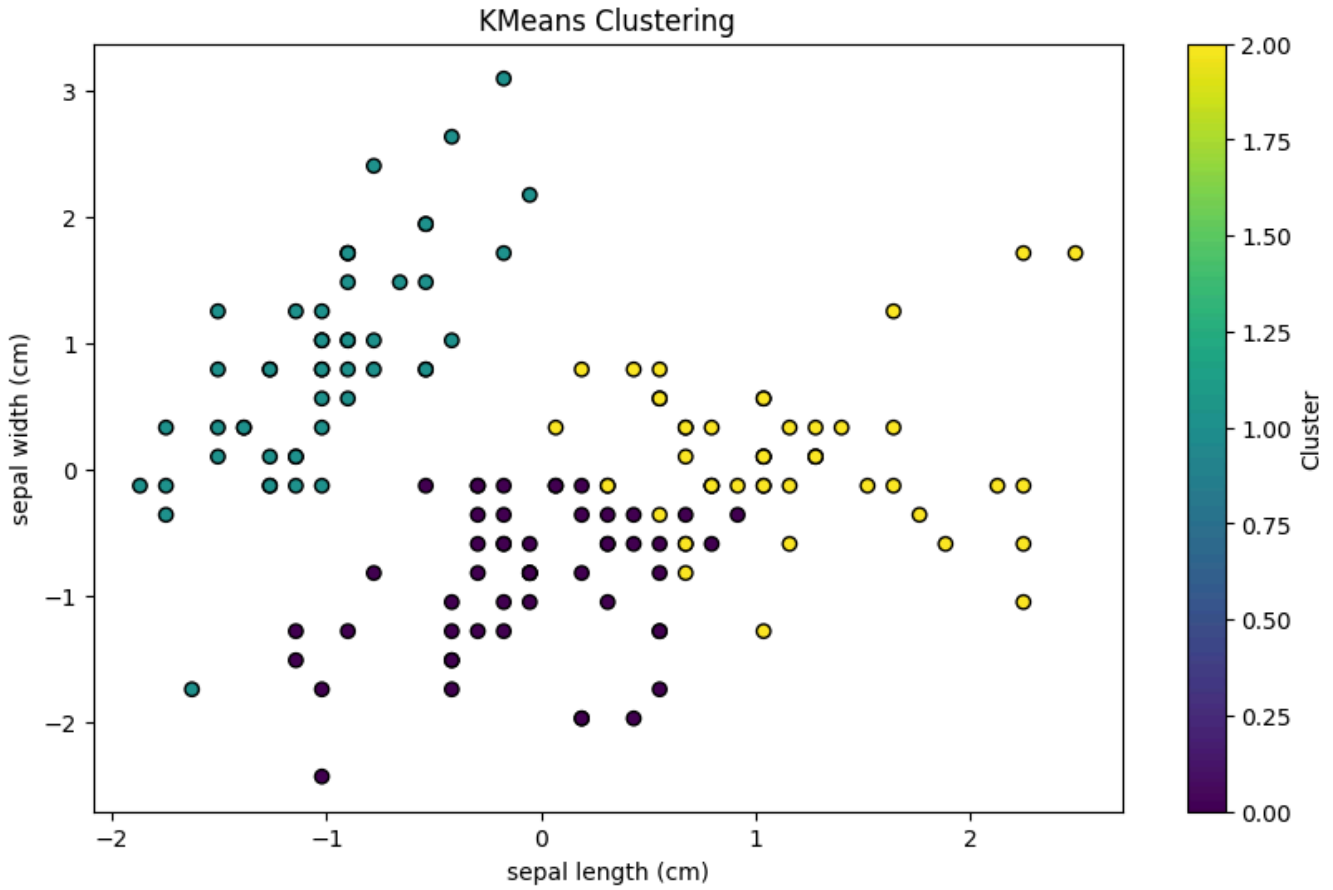
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Standardize the features
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Display the first few rows of the preprocessed data
print(data_scaled[:5])
```

```
[[-0.90068117  1.01900435 -1.34022653 -1.3154443 ]
 [-1.14301691 -0.13197948 -1.34022653 -1.3154443 ]
 [-1.38535265  0.32841405 -1.39706395 -1.3154443 ]
 [-1.50652052  0.09821729 -1.2833891  -1.3154443 ]
 [-1.02184904  1.24920112 -1.34022653 -1.3154443 ]]
```

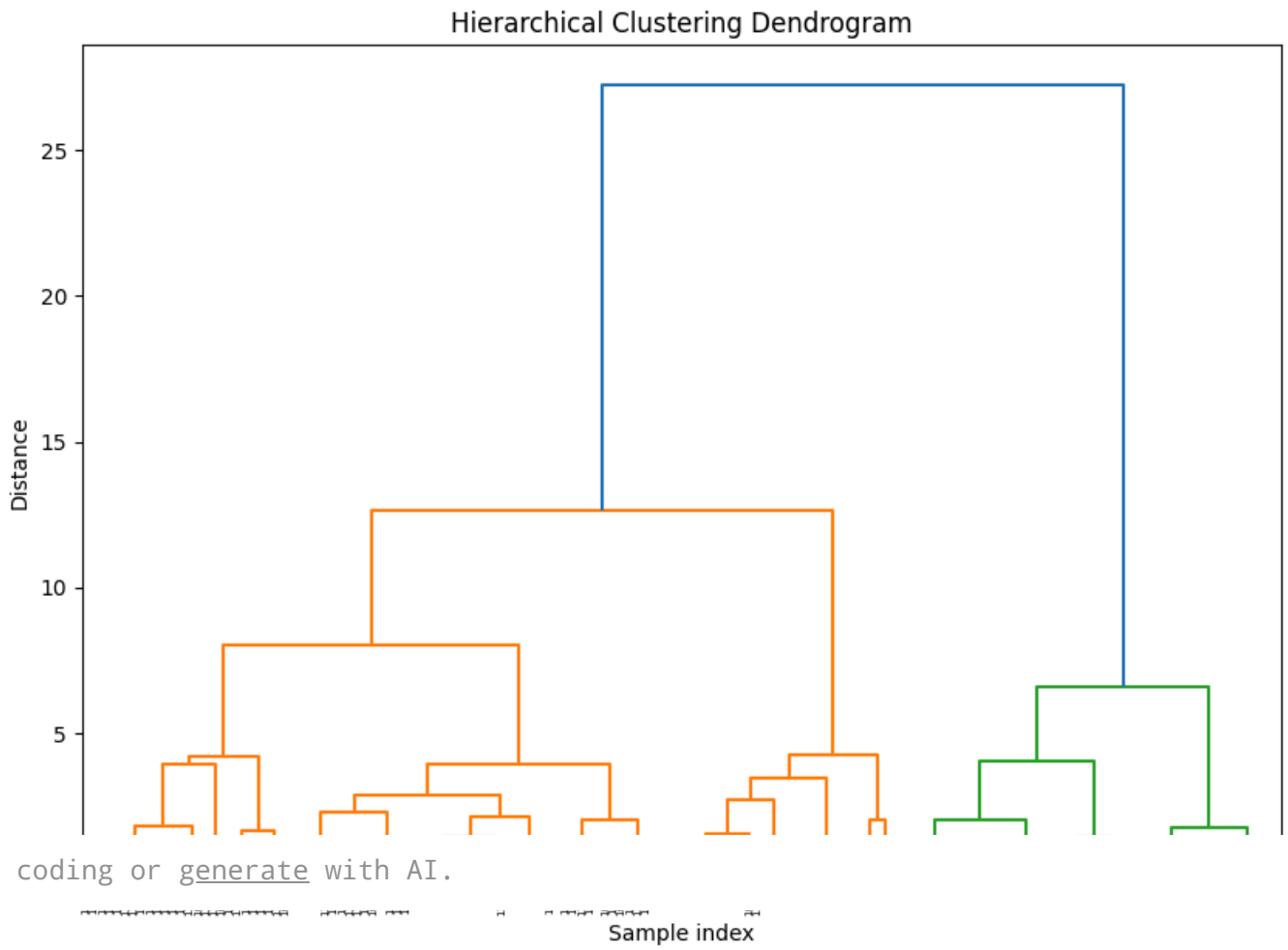
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Apply KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=0)  # Assuming we know there are 3 cluste
clusters = kmeans.fit_predict(data_scaled)

# Add the cluster assignments to the original data
data_with_clusters = pd.DataFrame(data_scaled, columns=iris.feature_names)
data_with_clusters['Cluster'] = clusters

# Visualize the clusters (using the first two features for simplicity)
plt.figure(figsize=(10, 6))
plt.scatter(data_with_clusters[iris.feature_names[0]], data_with_clusters[iris.featu
            c=data_with_clusters['Cluster'], cmap='viridis', marker='o', edgecolor='
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('KMeans Clustering')
plt.colorbar(label='Cluster')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureW
  super()._check_params_vs_input(X, default_n_init=10)
```




```
import scipy.cluster.hierarchy as sch

# Apply Hierarchical clustering
linked = sch.linkage(data_scaled, method='ward')

# Plot the dendrogram
plt.figure(figsize=(10, 7))
sch.dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_coun
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample index')
plt.ylabel('Distance')
plt.show()

# Assign clusters based on the dendrogram
clusters_hierarchical = sch.fcluster(linked, t=3, criterion='maxclust')  # Assuming

# Add the cluster assignments to the original data
```

```
data_with_clusters['Cluster_Hierarchical'] = clusters_hierarchical

# Visualize the clusters (using the first two features for simplicity)
plt.figure(figsize=(10, 6))
plt.scatter(data_with_clusters[iris.feature_names[0]], data_with_clusters[iris.featu
            c=data_with_clusters['Cluster_Hierarchical'], cmap='viridis', marker='o'
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('Hierarchical Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



Start coding or generate with AI.