Program 10: Organizing clusters as a hierarchical tree

Problem Definition:

The main objective of the problem is to evaluate hierarchical clustering on a subset of the Iris dataset. Select 10 samples from each species, totaling 30 samples. Use hierarchical clustering with the Ward method to create 3 clusters. Measure the clustering performance using three metrics: completeness score (to see if samples from the same species end up in the same cluster), silhouette score (to check how well-separated the clusters are), and Calinski-Harabasz score (to assess overall clustering quality). Create a dendrogram to visualize how the clusters are formed. This will show how well the clustering matches the true species labels and provide insight into the clustering's quality.

```
import pandas as pd  # Import pandas for data manipulation

from sklearn.preprocessing import StandardScaler  # Import StandardScaler for feature scaling

from sklearn.datasets import load_iris  # Import load_iris to fetch the Iris dataset

from sklearn.metrics import completeness_score, silhouette_score, calinski_harabasz_score  # Import clustering evaluation metrics

import scipy.cluster.hierarchy as sch  # Import hierarchical clustering functions from scipy

import matplotlib.pyplot as plt  # Import matplotlib for plotting


# Load the Iris dataset

iris = load_iris()  # Load the Iris dataset from sklearn


# Extract features and target

X = pd.DataFrame(iris.data, columns=iris.feature_names)  # Convert feature data to a pandas DataFrame with feature names

y = iris.target  # Extract target labels (species) from the dataset


# Combine features and target into a DataFrame

data = pd.concat([X, pd.Series(y, name='species')], axis=1)  # Combine features and target into a single DataFrame


# Sample 10 records from each category

sample = data.groupby('species').apply(lambda x: x.sample(10, random_state=42)).reset_index(drop=True)  # Sample 10 records per species from the combined DataFrame
```

```
# Separate features and target in the sample
X_sample = sample.drop(columns='species')  # Drop the 'species' column to get only features
y_sample = sample['species']  # Extract the 'species' column as the target

# Standardize the features
scaler = StandardScaler()  # Initialize StandardScaler for standardizing features
X_sample_scaled = scaler.fit_transform(X_sample)  # Fit and transform the features to have zero mean and unit variance

# Perform hierarchical clustering
linked = sch.linkage(X_sample_scaled, method='ward')  # Perform hierarchical clustering using the Ward method, which minimizes the variance of clusters

# Apply clustering to get cluster labels for a specific number of clusters (e.g., 3)
num_clusters = 3  # Set the number of clusters for the final clustering
labels = sch.fcluster(linked, num_clusters, criterion='maxclust')  # Form flat clusters from the hierarchical clustering

# Calculate evaluation metrics
completeness = completeness_score(y_sample, labels)  # Calculate the completeness score to evaluate how well the clusters match the true labels
silhouette = silhouette_score(X_sample_scaled, labels)  # Calculate the silhouette score to measure the clustering quality
calinski_harabasz = calinski_harabasz_score(X_sample_scaled, labels)  # Calculate the Calinski-Harabasz score for cluster validity

# Print evaluation metrics
print(f"Number of clusters: {num_clusters}")  # Print the number of clusters used in clustering
print(f"Completeness Score: {completeness:.2f}")  # Print the completeness score rounded to 2 decimal places
print(f"Silhouette Score: {silhouette:.2f}")  # Print the silhouette score rounded to 2 decimal places
```

```
print(f"Calinski-Harabasz Score: {calinski_harabasz:.2f}") # Print the Calinski-Harabasz score rounded to 2 decimal places


# Set up color mapping
species_colors = {i: color for i, color in enumerate(plt.cm.tab10(np.linspace(0, 1, len(np.unique(y_sample)))))} # Map each species to a unique color using the tab10 colormap


# Generate a dendrogram to visualize the hierarchical clustering
plt.figure(figsize=(12, 8)) # Create a figure with a specified size for the plot
dendrogram = sch.dendrogram(
    linked,
    orientation='top', # Arrange the dendrogram horizontally
    labels=y_sample.values, # Use the sampled target labels for the leaf labels
    distance_sort='descending', # Sort the dendrogram distances in descending order
    show_leaf_counts=True # Show the count of leaves in each cluster
)


# Add titles and labels
plt.title('Dendrogram of Hierarchical Clustering on Sample') # Set the title of the dendrogram plot
plt.xlabel('Sample Index') # Set the x-axis label
plt.ylabel('Euclidean Distance') # Set the y-axis label


plt.show() # Display the dendrogram plot
```