Hierarchical clustering

Hierarchical clustering is an unsupervised learning technique that builds a hierarchy of clusters, allowing for a comprehensive understanding of data structure. It is particularly useful when the number of clusters is not known a priori.

Key Features:

1. **Agglomerative Clustering:**
    - This bottom-up approach starts with each data point as its own cluster and iteratively merges the closest pairs based on a distance metric.
2. **Dendrogram Visualization:**
    - A dendrogram is a tree-like diagram that records the sequences of merges or splits. It helps in determining the optimal number of clusters by analyzing the distance at which clusters are merged.
3. **Scalability:**
    - Hierarchical clustering can be computationally intensive for large datasets, as its time complexity is typically $O(n^2)$. For very large datasets, alternative methods like DBSCAN or K-means may be more efficient.

How It Works:

1. **Data Preparation:**
    - Load the dataset and select relevant numerical features for clustering.
2. **Linkage Matrix Creation:**
    - Compute the pairwise distances between data points and use a linkage method (e.g., 'ward', 'single', 'complete') to build a linkage matrix.
3. **Dendrogram Plotting:**
    - Visualize the hierarchical relationships using a dendrogram to identify potential clusters.
4. **Cluster Assignment:**
    - Decide on the number of clusters by analyzing the dendrogram and assign each data point to a cluster.

Code Walkthrough:

1. **Data Loading and Feature Selection:**

```
 import pandas as pd
import numpy as np

# Load the dataset
data = pd.read_csv('your_dataset.csv')

# Select only numerical features for clustering
X = data.select_dtypes(include=[np.number])

# Display the first few rows
print(X.head())
```

2. **Linkage Matrix Creation:**

```
from scipy.cluster.hierarchy import linkage

# Create a linkage matrix using the 'ward' method
linked = linkage(X, method='ward')
```

3. **Dendrogram Plotting:**

```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram

# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=
plt.title('Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Euclidean Distance')
plt.show()
```

4. **Cluster Assignment:**

```
from sklearn.cluster import AgglomerativeClustering

# Initialize and fit the Agglomerative Clustering model
hierarchical = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage=
clusters = hierarchical.fit_predict(X)

# Add cluster labels to the original data
data['Cluster'] = clusters

# Display the first few rows with cluster labels
print(data.head())
```

Advantages:

- **No Need to Specify Number of Clusters:** Unlike K-means, hierarchical clustering does not require specifying the number of clusters in advance.
- **Hierarchical Structure:** The dendrogram provides a visual representation of the data's hierarchical structure, aiding in understanding the relationships between clusters.

Future Work:

- **Scalability Improvements:** Implementing more efficient algorithms or dimensionality reduction techniques to handle larger datasets.
- **Comparison with Other Clustering Methods:** Evaluating the performance of hierarchical clustering against other algorithms like K-means or DBSCAN to determine the most suitable method for specific applications.

References:

- Hierarchical Clustering with Scikit-Learn - GeeksforGeeks
- An Introduction to Hierarchical Clustering in Python - DataCamp
- Hierarchical Clustering with Python - AskPython

For a comprehensive understanding and additional examples, refer to the Hierarchical Clustering with Scikit-Learn - GeeksforGeeks tutorial.