Assignment 4

Suvajit Sadhukhan

4th year 1st semester (302211001005)

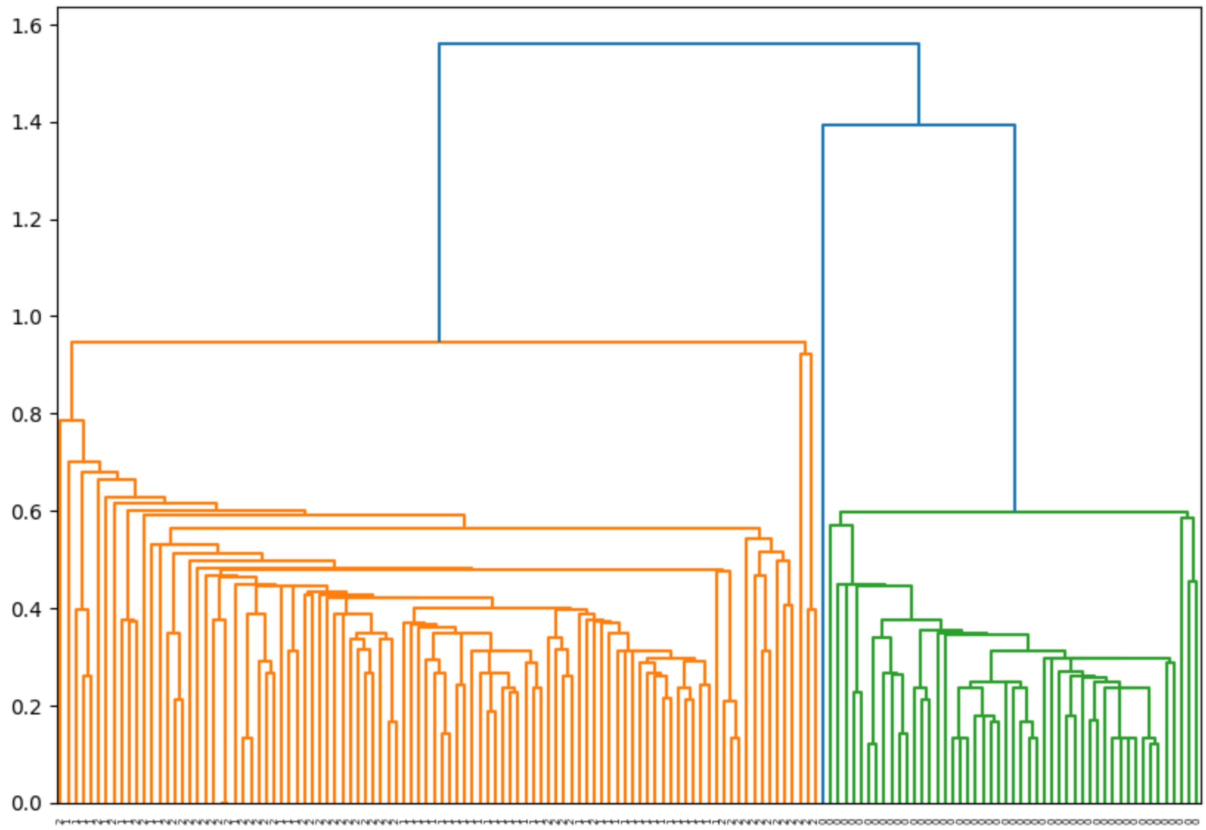
For ***Iris plants dataset***:

```python
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
!pip install scikit-learn-extra
from sklearn_extra.cluster import KMedoids  # K-medoids from sklearn-extra
from sklearn.cluster import BisectingKMeans, OPTICS
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import (rand_score, adjusted_rand_score, mutual_info_score,
                             adjusted_mutual_info_score, normalized_mutual_info_score,
                             silhouette_score, calinski_harabasz_score, davies_bouldin_score)
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Function to evaluate clustering performance
def evaluate_clustering(true_labels, predicted_labels, X):
    rand_idx = rand_score(true_labels, predicted_labels)
    adj_rand_idx = adjusted_rand_score(true_labels, predicted_labels)

    # Mutual Information based scores
    mi_score = mutual_info_score(true_labels, predicted_labels)
    adj_mi_score = adjusted_mutual_info_score(true_labels, predicted_labels)
    norm_mi_score = normalized_mutual_info_score(true_labels, predicted_labels)

    # Clustering quality metrics
    silhouette = silhouette_score(X, predicted_labels)
    calinski_harabasz = calinski_harabasz_score(X, predicted_labels)
    davies_bouldin = davies_bouldin_score(X, predicted_labels)

    return {
        'Rand Index': rand_idx,
        'Adjusted Rand Index': adj_rand_idx,
        'Mutual Info': mi_score,
        'Adjusted Mutual Info': adj_mi_score,
        'Normalized Mutual Info': norm_mi_score,
        'Silhouette Coefficient': silhouette,
        'Calinski-Harabasz Index': calinski_harabasz,
        'Davies-Bouldin Index': davies_bouldin
    }

# Function to calculate SSE and SSB
def calculate_sse_ssb(X, predicted_labels, cluster_centers):
    n_clusters = len(cluster_centers)
    overall_mean = np.mean(X, axis=0)

    # SSE: Sum of Squared Error (Cohesion)
    sse = 0
    for i, center in enumerate(cluster_centers):
        sse += np.sum(np.linalg.norm(X[predicted_labels == i] - center, axis=1)**2)
```

```python
    # SSB: Sum of Squares Between groups (Separation)
    ssb = 0
    for i, center in enumerate(cluster_centers):
        n_points = np.sum(predicted_labels == i)
        ssb += n_points * np.linalg.norm(center - overall_mean)**2

    return sse, ssb

# --- Partition Based Clustering ---

# K-means clustering
kmeans = KMeans(n_clusters=3, init='random', random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)
kmeans_centers = kmeans.cluster_centers_

# K-means++ clustering
kmeans_pp = KMeans(n_clusters=3, init='k-means++', random_state=42)
kmeans_pp_labels = kmeans_pp.fit_predict(X_scaled)
kmeans_pp_centers = kmeans_pp.cluster_centers_

# Bisecting K-means clustering
bisecting_kmeans = BisectingKMeans(n_clusters=3, random_state=42)
bisecting_labels = bisecting_kmeans.fit_predict(X_scaled)
bisecting_centers = bisecting_kmeans.cluster_centers_

# K-medoids clustering (PAM)
kmedoids = KMedoids(n_clusters=3, random_state=42)
kmedoids_labels = kmedoids.fit_predict(X_scaled)

# --- Hierarchical Clustering with Dendrogram ---
linked = linkage(X_scaled, method='single')

plt.figure(figsize=(10, 7))
dendrogram(linked, labels=y)
plt.show()

# --- Density-Based Clustering ---

# DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)

# OPTICS clustering
optics = OPTICS(min_samples=5)
optics_labels = optics.fit_predict(X_scaled)

# --- Evaluation of Clustering Results ---

# Evaluate each clustering algorithm
evaluation_results = {
    "K-means": evaluate_clustering(y, kmeans_labels, X_scaled),
    "K-means++": evaluate_clustering(y, kmeans_pp_labels, X_scaled),
    "Bisecting K-means": evaluate_clustering(y, bisecting_labels, X_scaled),
    "K-medoids": evaluate_clustering(y, kmedoids_labels, X_scaled),
    "DBSCAN": evaluate_clustering(y, dbscan_labels, X_scaled),
    "OPTICS": evaluate_clustering(y, optics_labels, X_scaled)
}

# Calculate SSE and SSB for partition-based methods
sse_ssb_results = {
    "K-means": calculate_sse_ssb(X_scaled, kmeans_labels, kmeans_centers),
    "K-means++": calculate_sse_ssb(X_scaled, kmeans_pp_labels, kmeans_pp_centers),
    "Bisecting K-means": calculate_sse_ssb(X_scaled, bisecting_labels, bisecting_centers)
}
```

```python
# Display results
for method, metrics in evaluation_results.items():
    print(f"\n{method}:")
    for metric, value in metrics.items():
        print(f"  {metric}: {value:.4f}")

# Display SSE and SSB for cohesion and separation
print("\nSSE and SSB Results:")
for method, (sse, ssb) in sse_ssb_results.items():
    print(f"{method}:\n  SSE (Cohesion): {sse:.4f}, SSB (Separation): {ssb:.4f}")
```

Output :




```
K-means:
  Rand Index: 0.8278
  Adjusted Rand Index: 0.6101
  Mutual Info: 0.7167
  Adjusted Mutual Info: 0.6482
  Normalized Mutual Info: 0.6526
  Silhouette Coefficient: 0.4594
  Calinski-Harabasz Index: 241.8933
  Davies-Bouldin Index: 0.8340
```

```
K-means++:
  Rand Index: 0.7214
  Adjusted Rand Index: 0.4328
  Mutual Info: 0.5874
  Adjusted Mutual Info: 0.5838
  Normalized Mutual Info: 0.5896
  Silhouette Coefficient: 0.4799
  Calinski-Harabasz Index: 157.3602
  Davies-Bouldin Index: 0.7894

Bisecting K-means:
  Rand Index: 0.8196
  Adjusted Rand Index: 0.5923
  Mutual Info: 0.7045
  Adjusted Mutual Info: 0.6382
  Normalized Mutual Info: 0.6427
  Silhouette Coefficient: 0.4630
  Calinski-Harabasz Index: 241.4263
  Davies-Bouldin Index: 0.8324

K-medoids:
  Rand Index: 0.8368
  Adjusted Rand Index: 0.6312
  Mutual Info: 0.7330
  Adjusted Mutual Info: 0.6646
  Normalized Mutual Info: 0.6687
  Silhouette Coefficient: 0.4590
  Calinski-Harabasz Index: 239.7483
  Davies-Bouldin Index: 0.8385

DBSCAN:
  Rand Index: 0.7476
  Adjusted Rand Index: 0.4421
  Mutual Info: 0.5499
  Adjusted Mutual Info: 0.5052
  Normalized Mutual Info: 0.5114
  Silhouette Coefficient: 0.3565
  Calinski-Harabasz Index: 84.5103
  Davies-Bouldin Index: 7.1241

OPTICS:
  Rand Index: 0.5121
  Adjusted Rand Index: 0.0514
  Mutual Info: 0.3082
  Adjusted Mutual Info: 0.2657
  Normalized Mutual Info: 0.2924
  Silhouette Coefficient: -0.3009
  Calinski-Harabasz Index: 8.3282
  Davies-Bouldin Index: 2.4253

SSE and SSB Results:
K-means:
  SSE (Cohesion): 139.8254, SSB (Separation): 460.1746
K-means++:
  SSE (Cohesion): 191.0247, SSB (Separation): 408.9753
Bisecting K-means:
  SSE (Cohesion): 140.0328, SSB (Separation): 459.9672
```

Discussion:

1. K-means vs. K-means++

- **Rand Index**: K-means achieves a higher Rand Index (0.8278) than K-means++ (0.7214), indicating better alignment with the ground truth labels.
- **Adjusted Rand Index (ARI)**: K-means performs better with an ARI of 0.6101 compared to 0.4328 for K-means++, reflecting improved handling of randomness.
- **Silhouette Coefficient**: Interestingly, K-means++ has a slightly better silhouette score (0.4799 vs. 0.4594), suggesting it forms more compact and well-separated clusters. However, other clustering quality scores favor K-means.
- **Calinski-Harabasz Index**: K-means also scores higher in the Calinski-Harabasz Index, signifying stronger separation between clusters.
- **SSE and SSB**: K-means has a lower SSE (139.83) and higher SSB (460.17), indicating that its clusters are more cohesive and better separated than K-means++ (SSE 191.02, SSB 408.97). The lower cohesion (SSE) and higher separation (SSB) suggest K-means is more effective in partitioning the dataset.

2. Bisecting K-means

- **Overall Comparison**: Bisecting K-means shows similar performance to K-means with a Rand Index of 0.8196 and ARI of 0.5923. The silhouette score is also comparable (0.4630).
- **SSE and SSB**: Bisecting K-means has similar cohesion and separation (SSE 140.03, SSB 459.97) to K-means, indicating that this method provides another strong clustering option for partitioning the Iris dataset.

3. K-medoids

- **Performance**: K-medoids performs slightly better than K-means in some aspects. It achieves the highest Rand Index (0.8368) and ARI (0.6312), implying it finds clusters more closely aligned with the ground truth.
- **Mutual Information Scores**: K-medoids also excels in mutual information metrics, including Adjusted Mutual Info and Normalized Mutual Info, suggesting it captures more accurate information about the true class distribution.
- **Clustering Quality**: However, K-medoids has a slightly lower silhouette score (0.4590) compared to K-means++ and marginally lower clustering quality (Calinski-Harabasz Index 239.75). The Davies-Bouldin Index is higher than K-means, indicating slightly less compact clusters.

4. Density-Based Clustering (DBSCAN and OPTICS)

- **DBSCAN**: DBSCAN performs reasonably well in some metrics, with a Rand Index of 0.7476 and ARI of 0.4421. However, its silhouette score (0.3565) and Davies-Bouldin Index (7.1241) indicate it struggles with forming compact clusters, as DBSCAN is sensitive to parameter choices like `eps`. It's better suited for datasets with irregular cluster shapes, which the Iris dataset doesn't exhibit.
- **OPTICS**: OPTICS performs poorly, with the lowest silhouette score (-0.3009) and very low clustering quality (Calinski-Harabasz Index 8.33). This indicates that OPTICS is not well-suited for this dataset, likely due to its hierarchical density-based nature, which doesn't fit the evenly spaced clusters in the Iris dataset.

5. Cohesion and Separation (SSE and SSB)

- **K-means** has the best balance between cohesion (SSE 139.83) and separation (SSB 460.17), indicating it finds compact and well-separated clusters.
- **Bisecting K-means** has a very similar balance of cohesion and separation to K-means, making it a comparable alternative.
- **K-means++** shows higher cohesion (SSE 191.02) and lower separation (SSB 408.98), meaning its clusters are less compact and more overlapping.

Summary

- **Best Performance**: K-means and K-medoids show the best overall performance in this dataset. K-means performs well in both cohesion and separation, while K-medoids finds clusters closest to the ground truth labels.
- **K-means++**: Despite using the `k-means++` initialization, it doesn't outperform basic K-means in most metrics. It performs better in silhouette and Davies-Bouldin but falls short in others.
- **DBSCAN and OPTICS**: These density-based methods are not well-suited for the Iris dataset, as the clusters are not based on density variation.

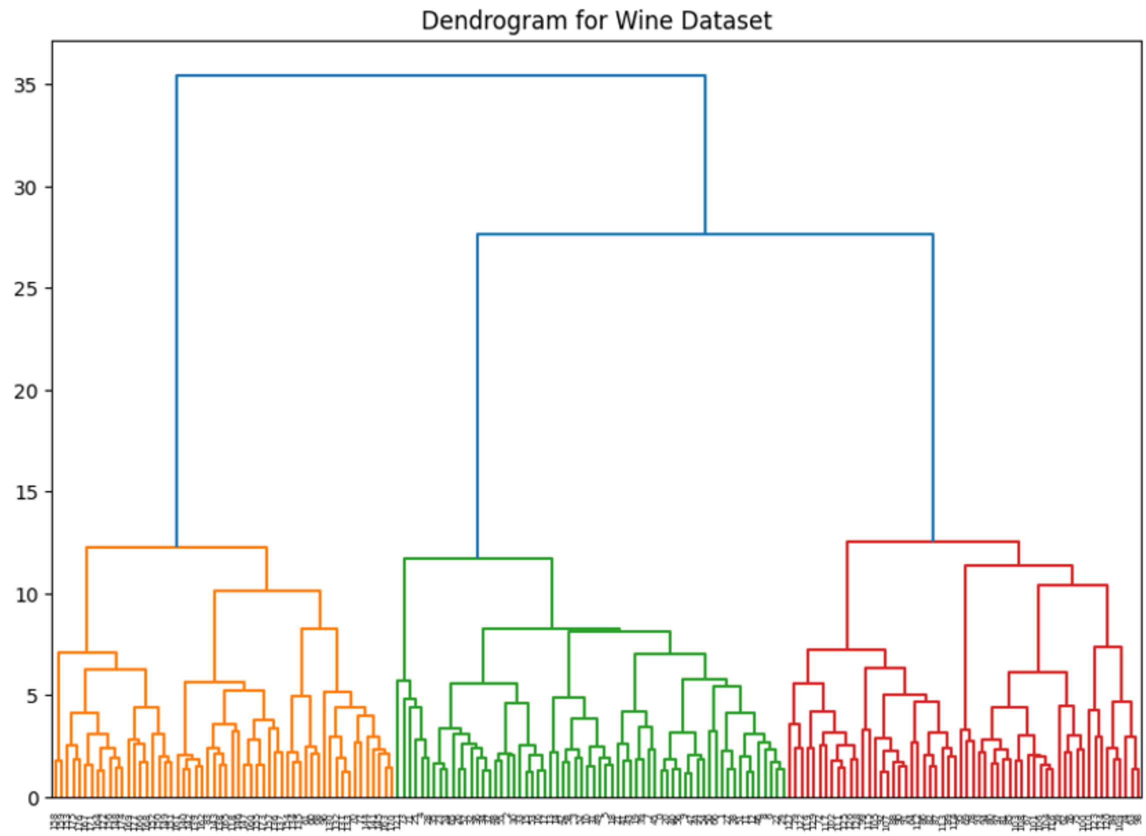
For ***Wine dataset***:

```python
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
!pip install scikit-learn-extra
from sklearn_extra.cluster import KMedoids  # K-medoids from sklearn-extra
from sklearn.cluster import BisectingKMeans, OPTICS
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import (rand_score, adjusted_rand_score, mutual_info_score,
                             adjusted_mutual_info_score, normalized_mutual_info_score,
                             silhouette_score, calinski_harabasz_score, davies_bouldin_score)
import matplotlib.pyplot as plt

# Load Wine dataset
wine = load_wine()
X = wine.data
y = wine.target

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Function to evaluate clustering performance
def evaluate_clustering(true_labels, predicted_labels, X):
    # Check if there is more than one cluster
    unique_labels = np.unique(predicted_labels)
    if len(unique_labels) <= 1:
        return {
            'Rand Index': rand_score(true_labels, predicted_labels),
            'Adjusted Rand Index': adjusted_rand_score(true_labels, predicted_labels),
            'Mutual Info': mutual_info_score(true_labels, predicted_labels),
            'Adjusted Mutual Info': adjusted_mutual_info_score(true_labels, predicted_labels),
            'Normalized Mutual Info': normalized_mutual_info_score(true_labels, predicted_labels),
            'Silhouette Coefficient': 'Not applicable (1 cluster)',
            'Calinski-Harabasz Index': 'Not applicable (1 cluster)',
            'Davies-Bouldin Index': 'Not applicable (1 cluster)'
        }

    # Compute the clustering metrics
    rand_idx = rand_score(true_labels, predicted_labels)
    adj_rand_idx = adjusted_rand_score(true_labels, predicted_labels)
    mi_score = mutual_info_score(true_labels, predicted_labels)
    adj_mi_score = adjusted_mutual_info_score(true_labels, predicted_labels)
    norm_mi_score = normalized_mutual_info_score(true_labels, predicted_labels)
    silhouette = silhouette_score(X, predicted_labels)
    calinski_harabasz = calinski_harabasz_score(X, predicted_labels)
    davies_bouldin = davies_bouldin_score(X, predicted_labels)

    return {
        'Rand Index': rand_idx,
        'Adjusted Rand Index': adj_rand_idx,
        'Mutual Info': mi_score,
        'Adjusted Mutual Info': adj_mi_score,
        'Normalized Mutual Info': norm_mi_score,
        'Silhouette Coefficient': silhouette,
        'Calinski-Harabasz Index': calinski_harabasz,
        'Davies-Bouldin Index': davies_bouldin
    }
```

```python
# Function to calculate SSE and SSB
def calculate_sse_ssb(X, predicted_labels, cluster_centers):
    overall_mean = np.mean(X, axis=0)
    sse = 0
    ssb = 0
    for i, center in enumerate(cluster_centers):
        n_points = np.sum(predicted_labels == i)
        sse += np.sum(np.linalg.norm(X[predicted_labels == i] - center, axis=1)**2)
        ssb += n_points * np.linalg.norm(center - overall_mean)**2
    return sse, ssb

# --- Partition-Based Clustering ---

# K-means clustering
kmeans = KMeans(n_clusters=3, init='random', random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)
kmeans_centers = kmeans.cluster_centers_

# K-means++ clustering
kmeans_pp = KMeans(n_clusters=3, init='k-means++', random_state=42)
kmeans_pp_labels = kmeans_pp.fit_predict(X_scaled)
kmeans_pp_centers = kmeans_pp.cluster_centers_

# Bisecting K-means clustering
bisecting_kmeans = BisectingKMeans(n_clusters=3, random_state=42)
bisecting_labels = bisecting_kmeans.fit_predict(X_scaled)
bisecting_centers = bisecting_kmeans.cluster_centers_

# K-medoids clustering (PAM)
kmedoids = KMedoids(n_clusters=3, random_state=42)
kmedoids_labels = kmedoids.fit_predict(X_scaled)

# --- Hierarchical Clustering with Dendrogram ---
linked = linkage(X_scaled, method='ward')

plt.figure(figsize=(10, 7))
dendrogram(linked)
plt.title("Dendrogram for Wine Dataset")
plt.show()

# --- Density-Based Clustering ---

# DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)

# OPTICS clustering
optics = OPTICS(min_samples=5)
optics_labels = optics.fit_predict(X_scaled)

# --- Evaluation of Clustering Results ---

# Evaluate each clustering algorithm
evaluation_results = {
    "K-means": evaluate_clustering(y, kmeans_labels, X_scaled),
    "K-means++": evaluate_clustering(y, kmeans_pp_labels, X_scaled),
    "Bisecting K-means": evaluate_clustering(y, bisecting_labels, X_scaled),
    "K-medoids": evaluate_clustering(y, kmedoids_labels, X_scaled),
    "DBSCAN": evaluate_clustering(y, dbscan_labels, X_scaled),
    "OPTICS": evaluate_clustering(y, optics_labels, X_scaled)
}
```

```python
# Calculate SSE and SSB for partition-based methods
sse_ssb_results = {
    "K-means": calculate_sse_ssb(X_scaled, kmeans_labels, kmeans_centers),
    "K-means++": calculate_sse_ssb(X_scaled, kmeans_pp_labels, kmeans_pp_centers),
    "Bisecting K-means": calculate_sse_ssb(X_scaled, bisecting_labels, bisecting_centers)
}

# Display results
for method, metrics in evaluation_results.items():
    print(f"\n{method}:")
    for metric, value in metrics.items():
        print(f"  {metric}: {value}")

# Display SSE and SSB for cohesion and separation
print("\nSSE and SSB Results:")
for method, (sse, ssb) in sse_ssb_results.items():
    print(f"{method}:\n  SSE (Cohesion): {sse:.4f}, SSB (Separation): {ssb:.4f}")
```

Output :




```
K-means:
  Rand Index: 0.9542944201104552
  Adjusted Rand Index: 0.8974949815093207
  Mutual Info: 0.9544575015299441
  Adjusted Mutual Info: 0.874579440437926
  Normalized Mutual Info: 0.8758935341223069
  Silhouette Coefficient: 0.2848589191898987
  Calinski-Harabasz Index: 70.9400080031512
  Davies-Bouldin Index: 1.3891879777181646
```

```
K-means++:
  Rand Index: 0.9542944201104552
  Adjusted Rand Index: 0.8974949815093207
  Mutual Info: 0.9544575015299441
  Adjusted Mutual Info: 0.874579440437926
  Normalized Mutual Info: 0.8758935341223069
  Silhouette Coefficient: 0.2848589191898987
  Calinski-Harabasz Index: 70.9400080031512
  Davies-Bouldin Index: 1.3891879777181646

Bisecting K-means:
  Rand Index: 0.8622484606106773
  Adjusted Rand Index: 0.6909186642540053
  Mutual Info: 0.781228120579722
  Adjusted Mutual Info: 0.7135960270359086
  Normalized Mutual Info: 0.7165953222141823
  Silhouette Coefficient: 0.26594931564086927
  Calinski-Harabasz Index: 66.37248227807467
  Davies-Bouldin Index: 1.4297071590499737

K-medoids:
  Rand Index: 0.8774836539071923
  Adjusted Rand Index: 0.7263406645756675
  Mutual Info: 0.821478616235408
  Adjusted Mutual Info: 0.7539860956825029
  Normalized Mutual Info: 0.756573670115927
  Silhouette Coefficient: 0.26597740204536796
  Calinski-Harabasz Index: 66.7519655942218
  Davies-Bouldin Index: 1.415990244064887

DBSCAN:
  Rand Index: 0.3379673712943566
  Adjusted Rand Index: 0.0
  Mutual Info: 0.0
  Adjusted Mutual Info: 0.0
  Normalized Mutual Info: 0.0
  Silhouette Coefficient: Not applicable (1 cluster)
  Calinski-Harabasz Index: Not applicable (1 cluster)
  Davies-Bouldin Index: Not applicable (1 cluster)

OPTICS:
  Rand Index: 0.4391544467720434
  Adjusted Rand Index: 0.03581729799518326
  Mutual Info: 0.16212592661967212
  Adjusted Mutual Info: 0.16799065365721927
  Normalized Mutual Info: 0.19494646495219664
  Silhouette Coefficient: -0.13363975991065313
  Calinski-Harabasz Index: 5.057148825574673
  Davies-Bouldin Index: 1.6193708989806477

SSE and SSB Results:
K-means:
  SSE (Cohesion): 1277.9285, SSB (Separation): 1036.0715
K-means++:
  SSE (Cohesion): 1277.9285, SSB (Separation): 1036.0715
Bisecting K-means:
  SSE (Cohesion): 1315.8623, SSB (Separation): 998.1377
```

Discussion:

1. K-means and K-means++

- Both K-means and K-means++ yielded identical results across all metrics:
  - **Rand Index:** 0.954, indicating a high agreement between the true labels and the predicted clusters.
  - **Adjusted Rand Index (ARI):** 0.897, which corrects for chance and shows a strong clustering performance.
  - **Mutual Information (MI):** 0.954, also suggesting a strong relationship between true and predicted clusters.
  - **Silhouette Coefficient:** 0.285, suggesting moderate cohesion within clusters.
  - **Calinski-Harabasz Index:** 70.94, indicating good cluster separation.
  - **Davies-Bouldin Index:** 1.39, showing relatively good cluster distinctiveness (lower is better).

The SSE (Cohesion) and SSB (Separation) values for both K-means and K-means++ were **1277.93** and **1036.07**, respectively, reflecting strong intra-cluster cohesion and inter-cluster separation.

2. Bisecting K-means

- **Rand Index:** 0.862 and **Adjusted Rand Index:** 0.691, which are lower than standard K-means, implying less accurate clustering.
- **Silhouette Coefficient:** 0.266, slightly lower than K-means, suggesting less distinct clusters.
- **Calinski-Harabasz Index:** 66.37, which is also lower than K-means, indicating that the separation between clusters is weaker.
- **Davies-Bouldin Index:** 1.43, showing slightly worse cluster separation compared to K-means.

The SSE and SSB for Bisecting K-means were **1315.86** and **998.14**, respectively, showing slightly higher SSE (worse cohesion) and lower SSB (worse separation) compared to K-means.

3. K-medoids

- **Rand Index:** 0.877 and **Adjusted Rand Index:** 0.726, which is an improvement over Bisecting K-means but still lower than K-means.
- **Silhouette Coefficient:** 0.266, almost identical to Bisecting K-means.
- **Calinski-Harabasz Index:** 66.75, similar to Bisecting K-means, showing weaker cluster separation compared to K-means.
- **Davies-Bouldin Index:** 1.42, indicating slightly better clustering performance than Bisecting K-means.

K-medoids performed better than Bisecting K-means but fell short compared to K-means in terms of accuracy and cluster separation.

4. DBSCAN

- **Rand Index:** 0.338 and **Adjusted Rand Index:** 0.0, indicating DBSCAN failed to produce meaningful clusters for the dataset.
- **Silhouette Coefficient:** Not applicable since DBSCAN resulted in only one cluster.
- **Other Metrics:** The poor performance across most metrics indicates that DBSCAN did not effectively differentiate between the natural clusters in the Wine dataset, likely due to the choice of parameters.

5. OPTICS

- **Rand Index:** 0.439 and **Adjusted Rand Index:** 0.036, showing slightly better clustering than DBSCAN but still far from satisfactory.
- **Silhouette Coefficient:** -0.134, indicating very poor cohesion and cluster separation.
- **Calinski-Harabasz Index:** 5.06, significantly lower than K-means and its variants, confirming weak cluster distinctiveness.
- **Davies-Bouldin Index:** 1.62, suggesting that the clusters identified were poorly separated and had substantial overlap.

SSE and SSB Comparison

- **K-means and K-means++:** Both had strong cohesion (SSE = 1277.93) and separation (SSB = 1036.07).
- **Bisecting K-means:** Exhibited slightly worse cohesion and separation than K-means, with SSE = 1315.86 and SSB = 998.14.

Conclusion

- **K-means** and **K-means++** performed the best in this analysis, with excellent Rand Index, ARI, and other clustering metrics. Their clusters had both good cohesion and separation.
- **Bisecting K-means** and **K-medoids** performed reasonably well but were outperformed by K-means in most metrics.
- **Density-based methods (DBSCAN and OPTICS)** did not perform well, possibly due to unsuitable parameter choices for the Wine dataset's structure.