Iris Dataset Clustering Analysis Report

Apply clustering algorithms to the Iris dataset to explore patterns and potentially identify natural groupings of iris flowers based on their measurements. (100 points)

The species of iris plants based on their features.
1. Iris Serosa
2. Iris Versicolor
3. Iris Virginica

NOTE: Refer to the ipynb file for code and graphs.

Imported necessary packages and loaded the Iris dataset

1. Load the and Explore the Dataset: (1 point)

a. Load the Iris dataset from the `sklearn.datasets` module

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Import packages
from sklearn.metrics import silhouette_score, pairwise_distances
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import MinMaxScaler
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN

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

b. Display basic information about the dataset (e.g., number of samples, features, target classes).

```
print("Number of Samples:", X.shape[0])
print("Number of Features:", X.shape[1])
print("Feature Names:", iris.feature_names)
print("Number of Classes:", len(np.unique(y)))
print("Target Classes:", iris.target_names)
species_counts = pd.Series(iris.target_names[iris.target]).value_counts()
print(", ".join(f"{index}: {value} samples" for index, value in species_counts.items()))
```

✓ 0.0s

```
Number of Samples: 150
Number of Features: 4
Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Number of Classes: 3
Target Classes: ['setosa' 'versicolor' 'virginica']
setosa: 50 samples, versicolor: 50 samples, virginica: 50 samples
```

2. Data Preprocessing: (assignment 2 code can be used) (1 point)

As a part of preprocessing, a check for missing values was made were no missing values found in the dataset [Note: Iris dataset has no missing values]. Although there were no missing values, a common step to remove any missing values data was performed.

```
df = pd.DataFrame(X, columns = iris.feature_names)
print("Missing Values:\n", df.isnull().sum()) # Missing values
df.dropna(inplace=True) # Remove missing values
preprocessed_df = df.values # Back to Numpy after preprocessing
```

✓ 0.0s

```
Missing Values:
 sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Scaling the features for better clustering performance.

```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Convert scaled data back to a DataFrame
scaled_iris_df = pd.DataFrame(X_scaled, columns=iris.feature_names)

# Display scaled data
print("After Standard Scaling:")
print(scaled_iris_df.head())
```

✓ 0.0s

```
After Standard Scaling:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0           0.222222          0.625000           0.067797          0.041667
1           0.166667          0.416667           0.067797          0.041667
2           0.111111          0.500000           0.050847          0.041667
3           0.083333          0.458333           0.084746          0.041667
4           0.194444          0.666667           0.067797          0.041667
```

Iris dataset is almost clean and doesn't contain any significant outliers. K-means and DBSCAN mainly rely on the Euclidean distance, therefore it will keep the features in the limited range.

3. Clustering Algorithms: (18 points for report))
e. Clustering algorithms to apply:

i. K-means Clustering

Assumptions
1. It assumes all clusters are spherical and isotropic (Radius is approximately equal in all directions).
2. All clusters are assumed to have the same variance.
3. The predetermined number of clusters (K) is known.

Advantages:
1. It is relatively simple to implement and computationally efficient.
2. It is simple, fast, scales really well to large datasets with high dimensionality.
3. It easily adapts to new examples without any issues.
4. It is easy to interpret and guarantees convergence.

Disadvantages:
1. It requires to pre-determine the number of clusters (k).
2. It is sensitive to outliers and noisy data, so it can't handle them very well.
3. It has a risk of getting stuck in local minima.
4. Poor performance on irregularly shaped clusters.

ii. Hierarchical clustering (Agglomerative clustering) - self learn

Assumptions:
1. The data naturally form nested clusters that resemble a tree.
2. Each data point can only belong to one cluster which is distinct, non-overlapping groups.
3. At the beginning, all points are assumed to be in a single cluster.

Advantages:
1. It doesn't require to specify the number of clusters in advance.
2. The resulting dendrogram is very informative, providing richer information and insight into a dataset.
3. It can adapt to non-spherical shapes unlike K-means.
4. Easy to use and implement.

Disadvantages:
1. It is computationally expensive, especially for larger datasets. The time complexity is O(n^2 log(n)). Computationally more intensive than k-means - mostly.
2. It never undo what was done previously, if incorrectly grouped it can't be corrected.
3. Can be sensitive to outliers and noise, affecting the resulting clusters.
4. Results are not replicable because it changes depending on the order of the data.

iii. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – selflearn

Assumptions:
1. In the data space, clusters are regions of high density which are separated by regions of lesser density.
2. In a cluster, not all are considered as points. Some are typically classified as noise and border points.
3. Does not have any shape assumptions.

Advantages:
1. It is robust to outliers and noise.
2. It doesn't require to specify the number of clusters in advance.
3. It only requires two parameters: epsilon and min points
4. Can identify clusters of any shape.

Disadvantages:
1. The values of eps, min_samples and the distance metric used have substantial effects on the clustering quality.
2. It doesn't cluster data sets well when there are large differences in density.
3. It doesn't perform well on datasets with categorical features.
4. It struggles with high dimensionality data.

4. Clustering Experimentation: (20 + 10 (report) points)

f. Apply each clustering algorithm to the preprocessed dataset.

Applying **K-means** on the preprocessed dataset:

```
k_values = [2, 3, 4, 5, 6]
k_means_results = {}
inertia_scores = {}

for k in k_values:
    k_means = KMeans(n_clusters = k, random_state = 42)
    k_means_labels = k_means.fit_predict(X_scaled)
    k_means_results[k] = k_means_labels
    inertia_scores[k] = k_means.inertia_
```

```
kmeans_silhouette_scores = {}

for k, labels in k_means_results.items():
    silhouette_score_kmeans = silhouette_score(X_scaled, labels)
    kmeans_silhouette_scores[k] = silhouette_score_kmeans

print("K-Means Silhouette Scores for five k values:\n")
for k in k_values:
    print(f"\nk = {k}: Silhouette Score = {kmeans_silhouette_scores[k]:.3f}")
    print(f"K-Means inertia = {inertia_scores[k]:.3f}")
```

Applying **Hierarchical clustering (Agglomerative)** on the preprocessed dataset:

```
linkage = ['single', 'average', 'complete', 'ward']
hier_results = {}

for method in linkage:
    hier = AgglomerativeClustering(n_clusters=3, linkage=method)
    hier_labels = hier.fit_predict(X_scaled)
    hier_results[method] = hier_labels
```

```
hier_silhouette_scores_ = {}

for method, labels in hier_results.items():
    score = silhouette_score(X_scaled, labels)
    hier_silhouette_scores_[method] = score

print("Agglomerative Silhouette Scores for different linkages:\n")
for method in linkage:
    print(f"Linkage = {method}: Silhouette Score = {hier_silhouette_scores_[method]:.3f}")
```

Applying **DBSCAN** on the preprocessed dataset:

```
eps_values = np.arange(0.1, 0.6, 0.1)
min_values = range(2, 7)
dbscan_results = {}

for eps in eps_values:
    for min_samples in min_values:
        dbscan = DBSCAN(eps = eps, min_samples = min_samples)
        dbscan_labels = dbscan.fit_predict(X_scaled)
        dbscan_results[(eps, min_samples)] = dbscan_labels
```

```
dbscan_silhouette_scores_ = {}

for params, labels in dbscan_results.items():
    if len(set(labels)) > 1:
        score = silhouette_score(X_scaled, labels)
        dbscan_silhouette_scores_[params] = score

print("DBSCAN Silhouette Scores with different eps and min samples:\n")
for params in dbscan_results:
    if params in dbscan_silhouette_scores_:
        print(f"eps = {params[0]:.1f}, min_samples = {params[1]}: Silhouette Score = {dbscan_silhouette_scores_[params]:.3f}")
```

g. Explore different parameter settings (e.g., number of clusters for K-means, linkage method for hierarchical clustering, epsilon and min_samples for DBSCAN). Write your observations in report

Exploring parameter settings:

K-means Clustering: When implementing K-means, five different number of cluster k values were implemented. These k values are: 2, 3, 4, 5, 6

Hierarchical clustering (Agglomerative): When implementing Agglomerative Clustering, 4 various linkage methods were tested. These linkage methods are Single, Average, Complete, Ward.

DBSCAN: When implementing DBSCAN, various eps and min_samples values were tested. The eps values of 0.1, 0.2, 0.3, 0.4 and min_samples values of 2, 3, 4, 5, 6 were used in DBSCAN.

5. Evaluation: (10 + 10 points)

h. Evaluate the quality of clustering using metrics like silhouette score (self-learn, you can also use inertia and Dunn index taught in class) and visual inspection:

i. Silhouette Scores: A higher score indicates better-defined clusters.

K-means Clustering (with inertia):

```
K-Means Silhouette Scores and inertia for five k values:

k = 2: Silhouette Score = 0.630
K-Means inertia = 12.128

k = 3: Silhouette Score = 0.505
K-Means inertia = 6.982

k = 4: Silhouette Score = 0.445
K-Means inertia = 5.517

k = 5: Silhouette Score = 0.353
K-Means inertia = 4.581

k = 6: Silhouette Score = 0.355
K-Means inertia = 3.976
```

Observations:

1. The silhouette score is highest **(0.630)** when the number of clusters **k = 2**, suggesting that two clusters provide the best separation. The lowest silhouette score of **0.353** is observed when **k = 5** which close to **0.355** which **k = 6** has.
2. As k increases, the silhouette score decreases. This might be because these additional clusters are less distinct or more overlapping. **k = 3** is more appropriate, since there are 3 species in the iris data set. Even though the score is high, it's not the highest.
3. As k increases, inertia value decreases indicating the clusters are more compact. One with a low inertia means the model is good; as K increases, inertia decreases. Again, **k = 3** is more appropriate with inertia of **6.982**.
4. K-means clustering may yield the best clusters with **k = 2** or **k = 3**. This is because with iris dataset, we know the number of species.

Hierarchical clustering (Agglomerative)

```
Agglomerative Silhouette Scores for different linkages:

Linkage = single: Silhouette Score = 0.532
Linkage = average: Silhouette Score = 0.505
Linkage = complete: Silhouette Score = 0.504
Linkage = ward: Silhouette Score = 0.505
```

Observations:

1. The Silhouette Score is relatively similar across different linkages. average and ward with a score **0.505**, perform slightly better than the other two linkages. Even though single has the highest score, it is susceptible to noise and outliers.
2. The **ward** creates clusters of same size, indicating it performs well at capturing the natural groupings in the Iris dataset.

DBSCAN:

```
DBSCAN Silhouette Scores with different eps and min samples:

eps = 0.1, min_samples = 2: Silhouette Score = 0.230
eps = 0.1, min_samples = 3: Silhouette Score = 0.203
eps = 0.1, min_samples = 4: Silhouette Score = 0.134
eps = 0.1, min_samples = 5: Silhouette Score = 0.057
eps = 0.1, min_samples = 6: Silhouette Score = 0.057
eps = 0.2, min_samples = 2: Silhouette Score = 0.390
eps = 0.2, min_samples = 3: Silhouette Score = 0.556
eps = 0.2, min_samples = 4: Silhouette Score = 0.556
eps = 0.2, min_samples = 5: Silhouette Score = 0.556
eps = 0.2, min_samples = 6: Silhouette Score = 0.556
eps = 0.3, min_samples = 2: Silhouette Score = 0.630
eps = 0.3, min_samples = 3: Silhouette Score = 0.630
eps = 0.3, min_samples = 4: Silhouette Score = 0.468
eps = 0.3, min_samples = 5: Silhouette Score = 0.468
eps = 0.3, min_samples = 6: Silhouette Score = 0.468
eps = 0.4, min_samples = 2: Silhouette Score = 0.630
eps = 0.4, min_samples = 3: Silhouette Score = 0.630
eps = 0.4, min_samples = 4: Silhouette Score = 0.630
eps = 0.4, min_samples = 5: Silhouette Score = 0.630
eps = 0.4, min_samples = 6: Silhouette Score = 0.630
```

Observations:

1. The Silhouette Score is highest **(0.630)** when **eps = 0.3**, **min_samples = 2, 3** and **eps = 0.4**, **min_samples = 2, 3, 4, 5, 6**. This shows that it has a good balance between finding dense clusters while minimizing the number of outliers.
2. The silhouette scores are lower when **eps = 0.1**, indicating many points being considered as noise or clusters.
3. The Silhouette Score stays consistent **(0.630)** when **eps = 0.4** and all values of min_samples, showing the clusters are well separated and stable.
4. It shows the worst Silhouette Score of **0.057** when **eps = 0.1** and **min_samples = 5, 6**

Dunn Index:

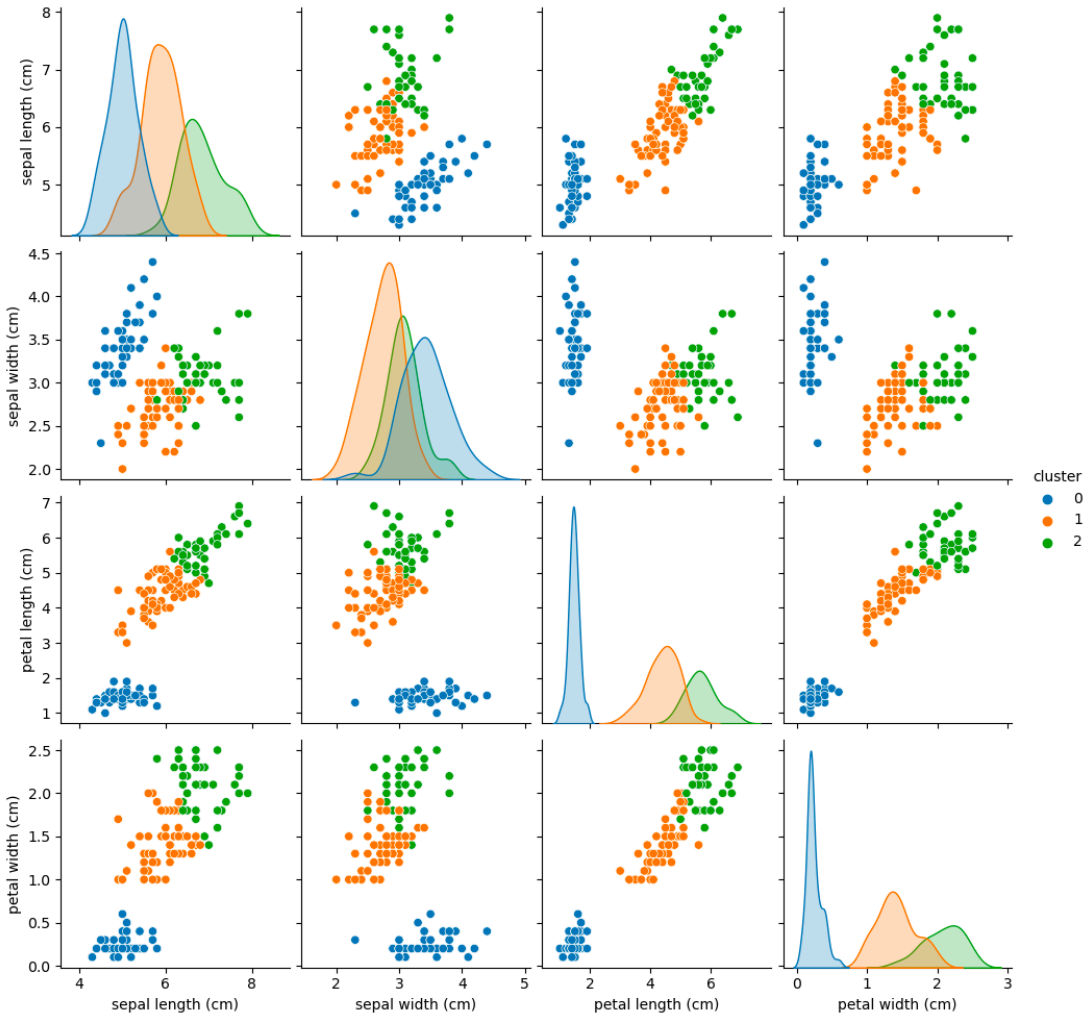
```
Dunn Index for K-Means: 0.069
Dunn Index for Hierarchical: 0.113
Dunn Index for DBSCAN with eps = 0.3 and min_samples = 2: 0.358
```

Observations:
1. Dunn index is used to identify dense and well-separated groups. It is the ratio between minimum inter-cluster distance and maximum intra-cluster distance. The higher the Dunn Index value, the better the clustering results.
2. K-means has a Dunn index of **0.069**, which is relatively low. This shows that there is more overlap between the clusters, suggesting that they are more dispersed or closer to each other.
3. Hierarchical has slightly higher Dunn index of **0.113**. This shows that there is an increase in the separation of the clusters compared to K-means.
4. DBSCAN showed the highest Silhouette Scores when **eps = 0.3** and **min_samples = 2, 3** without noises. It show significantly higher Dunn index of **0.358** when the same eps and min_samples values were used. This shows a much better cluster separation and compactness indicating DBSCAN performed better in terms of clustering quality.

j. Visualize clusters using scatter plots or other appropriate visualizations. Write observations along with visualization in report.

<u>Pairwise Plot:</u>

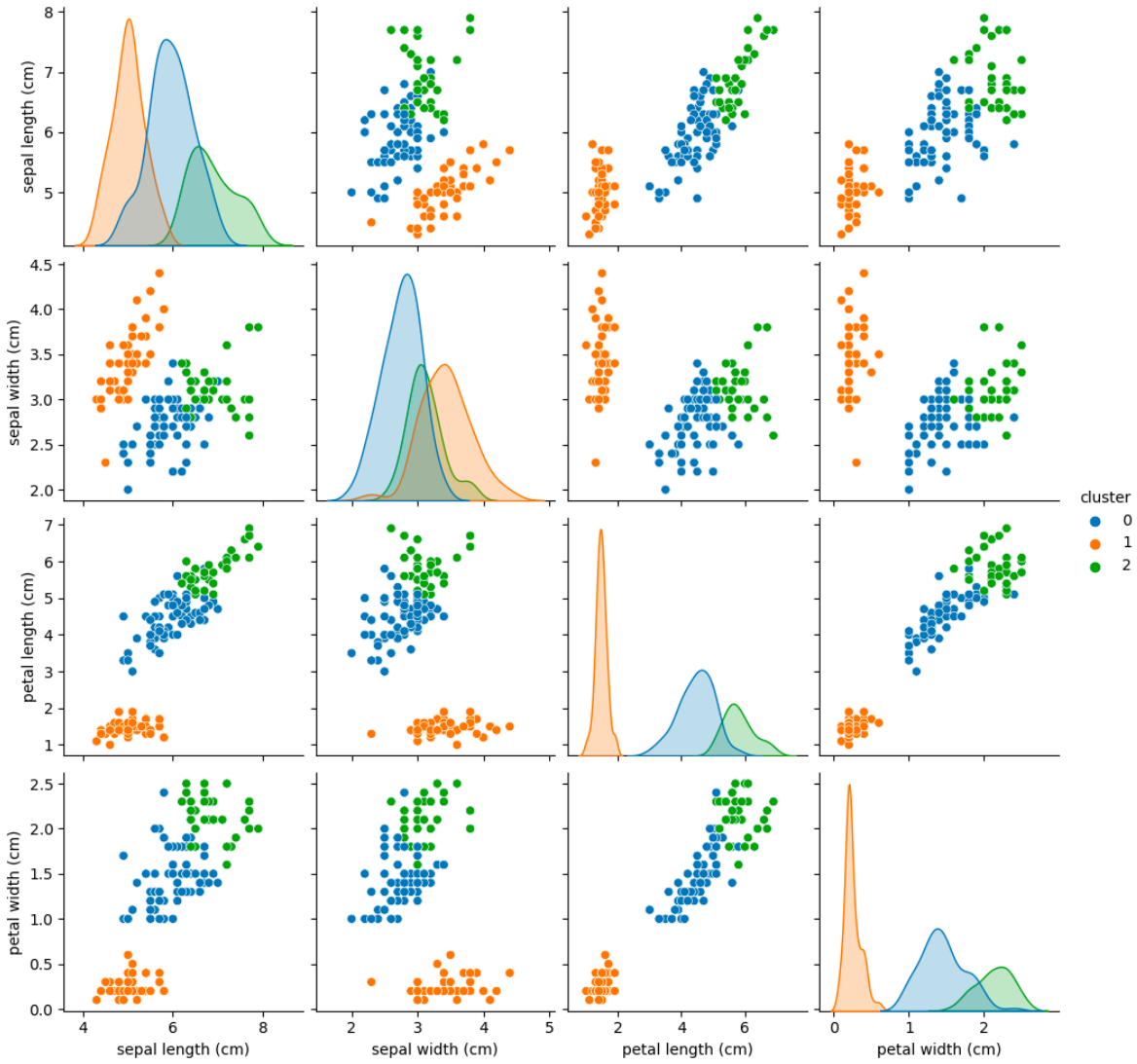
K-means Clustering Visualization:



Observations:

1. When we observe the petal length and petal width plot, the clusters are separated very well suggesting that these features are good predictors for differentiating the Iris species in the dataset.
2. The clusters in sepal length and width are more spread out with some overlap, indicating less discriminative for clustering in this dataset.
3. Cluster 0 has smaller petal lengths and widths, suggesting it is setosa species. The other two are Versicolor and Virginica
4. The density plots for petal length and petal width are multi-modal because of their distinct groupings. It is more uni-modal or overlapping for sepal width and sepal length.

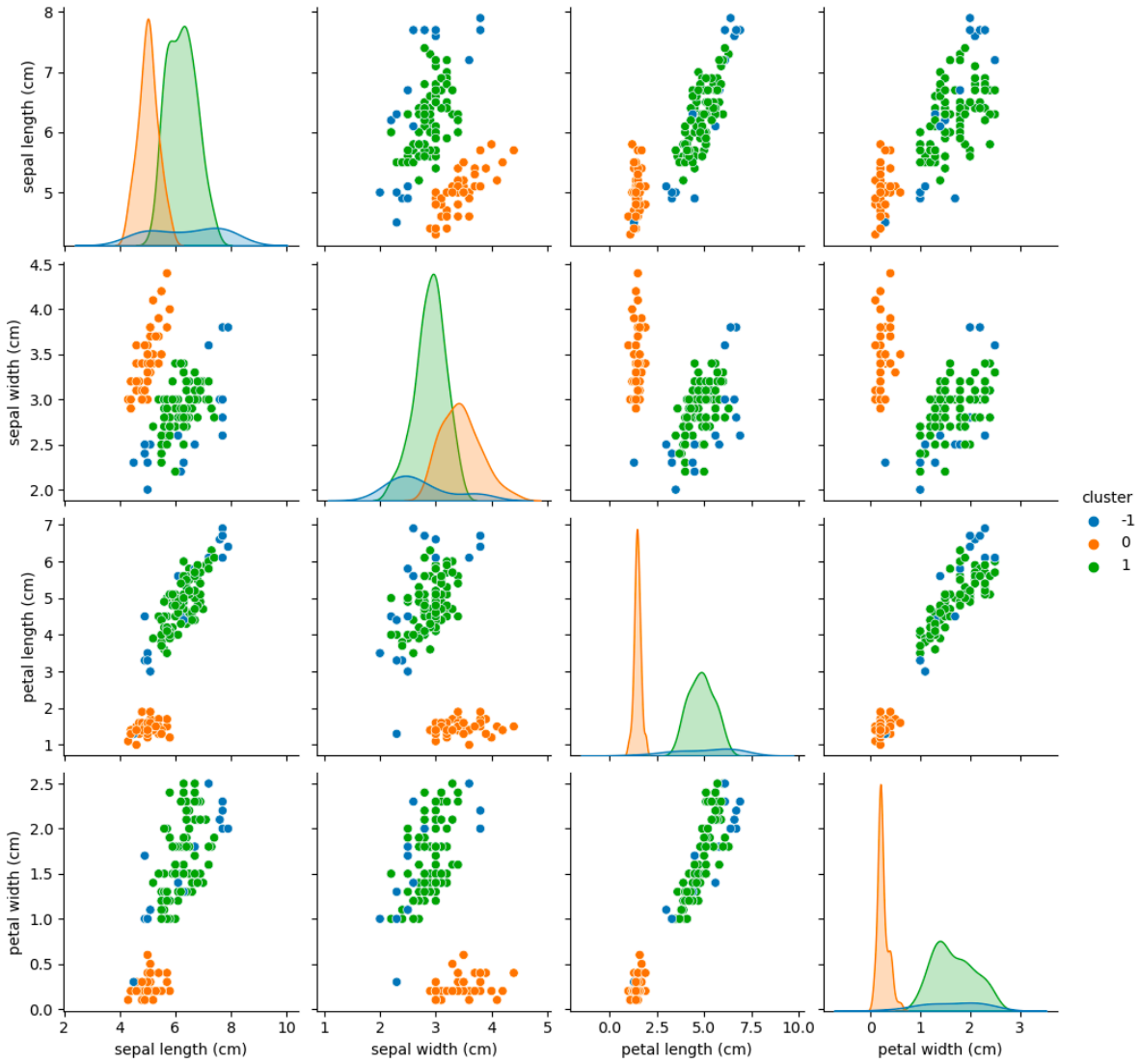
Agglomerative Clustering Visualization:



Observations:

1. Similar to K-means, the separation is clear in the scatter plots that has petal length and petal width.
2. The clusters based on petal features looks more informative than sepal measurements for distinguishing the species.
3. The density plots on the diagonal shows that the green cluster tends to have smaller values for petal length and petal width.
4. There is a noticeable overlap in sepal width across all clusters.
5. Both K-means Clustering and Agglomerative Clustering show similar patterns of clustering, especially in petal measurements.

DBSCAN Visualization:



Observations:
1. The pair plot uses min_samples = 5 and eps = 0.5 to visualize clusters.
2. The two dense clusters suggest DBSCAN has detected two dense regions in the data space.
3. The blue cluster is denser and tightly packed indicating the cluster is well defined.
4. The orange cluster more spread out compared to blue cluster. This is a less dense region with several smaller sub-cluster that DBSCAN could not differentiate with given eps value.
5. Certain criteria clearly distinguish across clusters, others have a lot of overlap. This means that some characteristics might provide more information about a cluster than others.
6. The two distributions show varying levels of separation based on distinct features in certain density plot, while in others there is a significant overlap. By the looks of it, the eps value worked well, as it has identified distinct clusters without too much scatter or noise.

6. Interpretation and Analysis: (10 + 10 points)

k. Analyze the clustering results:
l. Interpret the identified clusters and discuss their characteristics.
m. Compare and contrast the effectiveness of different clustering algorithms for this dataset.
Write observations along with visualization in report.

Note: Parts 4, 5, 6, and 7 are similar, so the graphs and observations are discussed throughout the report. Please refer to the graphs and observations both above and below as applicable.

7. Visualization: (10 points)

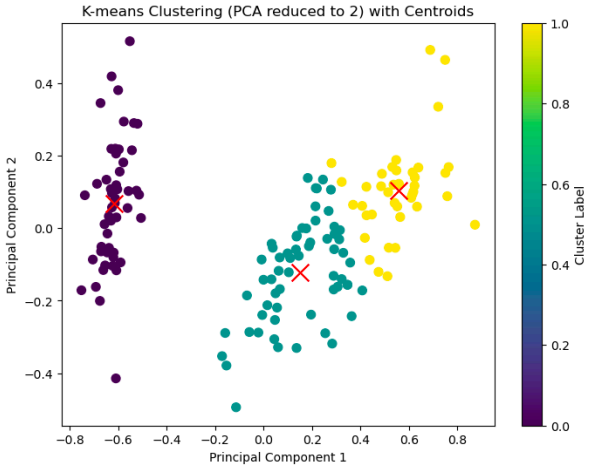
n. Visualize the clustered data:

o. Plot clusters using scatter plots, with each cluster represented by a different color.

p. Optionally, visualize centroids or dendrograms (for hierarchical clustering).

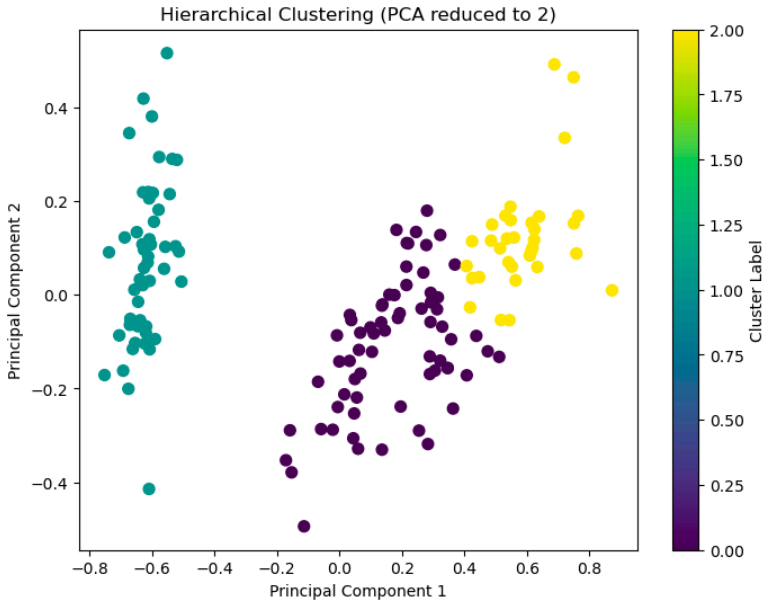
The data points reduced to two dimensions using Principal Component Analysis (PCA)

K-means with centroids:



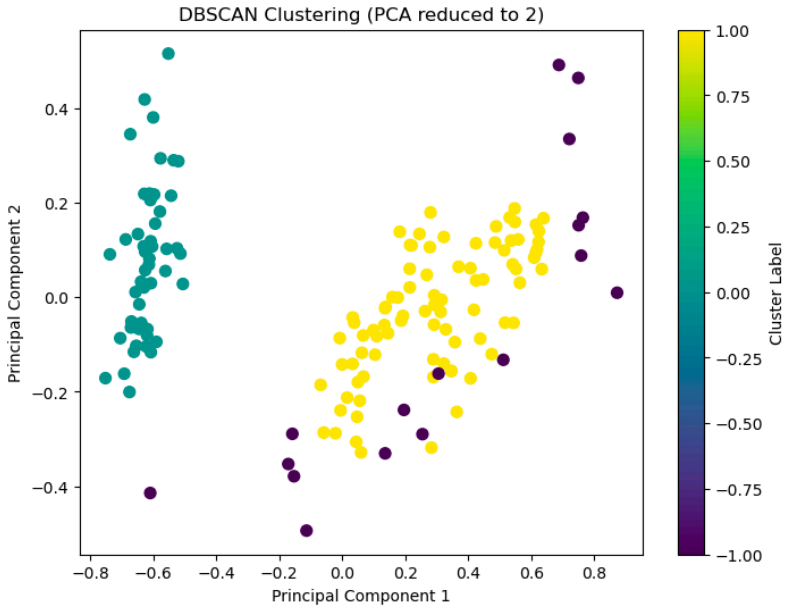
Observations:
1. The plot represents a K-means clustering with the centroids marked by red crosses. There are three unique clusters that are each shown by a different color.
2. It can be seen that the K-means has successfully reduced the in-cluster variation because the centroids are centered within their respective clusters. The components are separated after dimensionality reduction by PCA.
3. The clusters in the plot are separated well, indicating K-means identified the clear boundary between the groups.
4. It can be seen that there are not any notable outliers that diverge from their clusters' overall distribution, showing that the K-means did a good job of capturing the primary structure of the data.
5. The dispersion of points in each cluster is different. Some of the points are tightly packed and some are spread out which showcases the diversity in the data distribution captured.
6. The clusters are based on standardized features, which provides a uniform metric for comparison.

Hierarchical Clustering:



Observations:
1. The plot represents a Hierarchical clustering model. There are three unique clusters that are each shown by a different color in the dimensionality reduction by PCA.
2. Unlike k-means, which divides data into distinct clusters, hierarchical clustering produces a tree of clusters, providing several complexity levels. The clusters display a structure similar to a chain, which is indicative of the linkage used.
3. The method's nested nature can be seen in a seamless change between clusters, which indicates a hierarchy in which some clusters are parts of larger clusters.
4. Hierarchical clustering does not produce clusters that are evenly separated showing its flexibility in identifying groups that occur naturally. It’s not the same with K-means clustering.
5. It grouped the data points based on closeness and connection without assigning any pre-defined cluster shapes.

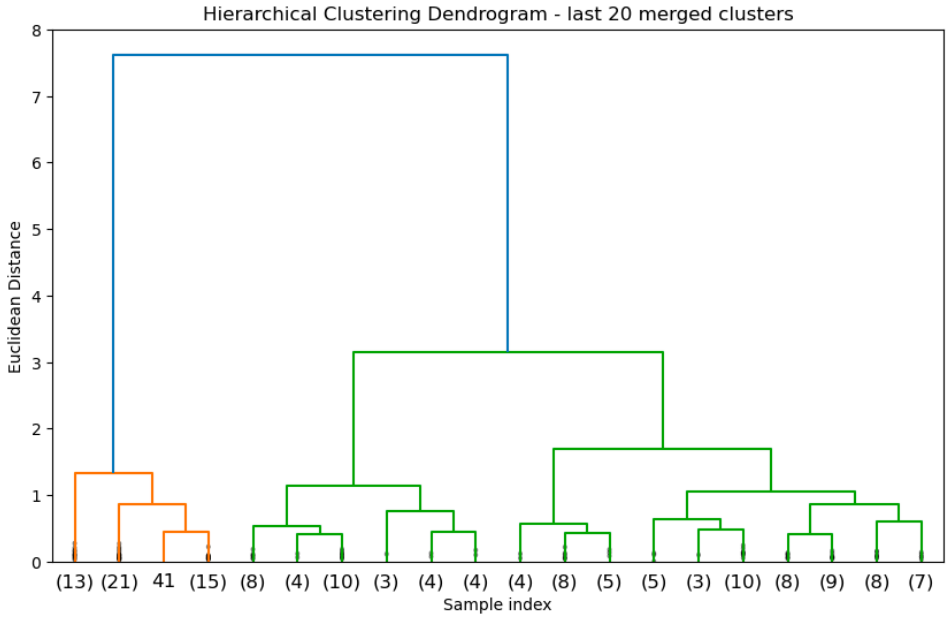
DBSCAN:



Observations:

1. This DBSCAN does not require a pre-defined number of clusters. It has identified two main clusters with some noise points.
2. It efficiently classifies low points as noise or outliers and isolates dense areas as clusters offering robustness against noise and outliers. Density is what distinguishes the clusters created.
3. The clustering in the plot shows the ability of DBSCAN to handle different cluster shapes with non-spherical and unevenly shaped clusters.
4. It can be seen that the clusters with different densities are present which shows the flexibility of DBSCAN. It does not force data points into existing clusters.

Dendrogram Hierarchical Clustering:



Observations:
1. The dendrogram shows the last 20 merged clusters. The hierarchical clustering has found several clusters at different levels of distance, indicating the dissimilarity between clusters. The 3 cluster groups shown align perfectly with three species in the Iris dataset. It offers insights into cluster hierarchies.
2. The hierarchical nature of the data linkages is captured by the clusters' differences in size and shape, which is a feature of hierarchical clustering.
3. The heights of the vertical lines shows the distance at which the clusters were merged. Taller the lines, more dissimilar clusters being merged.
4. The distance limit that keeps clusters from merging is shown by the horizontal lines. Observing this dendrogram, it can be seen that the green line clusters merge at a lower height (closer distance), indicating greater similarity within that group.
5. At smaller distances, the clustering appears to be smoother with numerous distinct merges, showing a complex cluster structure in the data.
6. The big blue vertical line that shows a merge at a significantly higher distance, indicating a more noticeable distinction between the two bigger clusters it connected.

Comparison of Clustering Algorithms:
1. When the number of clusters k = 3 was selected, K-means performed effectively matching the number of Iris species. However, its Dunn index was lower which indicating it has less significant distinction.
2. Hierarchical Clustering showed consistent results across different linkages. The ward linkage balanced cluster size well. The Dunn index indicated moderate separation between clusters.
3. DBSCAN showed robust results in identifying dense clusters and handling noise. It was able to achieve the highest silhouette scores and Dunn index with optimal eps and min_samples eps=0.3, min_samples=2, 3) showing compactness and a good cluster separation. It effectively found clusters with different density.
4. K-means and Hierarchical Clustering produced good results, but they are more sensitive to outliers and noise compared to DBSCAN.
5. DBSCAN is highly sensitive to the eps and min_samples values, but it does not require specifying the number of clusters which gives flexibility. On the other hand, K-means require specifying them. In this case, it is Iris dataset so we know k = 3 can be ideal, but with other datasets we might know what k value to choose.
6. Choosing the right clustering method depends on the understanding of each algorithm's strengths and weaknesses in relation to the dataset characteristics. DBSCAN is doing a good job identifying and separating noise from clusters.

References:

1. https://www.geeksforgeeks.org/demonstration-of-k-means-assumptions/#
2. https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages
3. https://datarundown.com/k-means-clustering-pros-cons/
4. https://online.stat.psu.edu/stat505/lesson/14/14.4
5. https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/
6. https://ml-explained.com/blog/dbscan-explained

7. https://en.wikipedia.org/wiki/DBSCAN
8. https://www.educative.io/answers/what-is-silhouette-score
9. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
10. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html
11. https://www.geeksforgeeks.org/implementing-agglomerative-clustering-using-sklearn/
12. https://hex.tech/blog/comparing-density-based-methods/
13. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
14.https://www.tutorialspoint.com/machine_learning_with_python/clustering_algorithms_overview.htm