Lesson 11 Assignment

July 21, 2024

[1]:
```
#!/usr/bin/env python
```

0.0.1 Task:

Data Set Information

The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa, and Canadian, 70 elements each, randomly selected for the experiment. High-quality visualization of the internal kernel structure was detected using a soft X-ray technique. It is non-destructive and considerably cheaper than other more sophisticated imaging techniques like scanning microscopy or laser technology. The images were recorded on 13x18 cm X-ray KODAK plates. Studies were conducted using combined harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

Attribute Information

- To construct the data, seven geometric parameters of wheat kernels were measured:

1. area A,
2. perimeter P,
3. compactness C = $4piA/P^2$,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove.

Please use this data to finish the following tasks. 1. Explore the data set. (10 points) 2. Use K-means clustering to group the seed data. (30 points) 3. Use different linkage type for Hierarchical clustering to the seed data, which linkage type give the best result? (30 points) 4. Use DBscan clustering group the seed data and find the best epses and min_samples value. (30 points)

[69]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
from scipy.spatial import ConvexHull
from tabulate import tabulate

def load_data(file_path):
    """
    Load the dataset from a given file path.

    Parameters:
    file_path (str): Path to the CSV file containing the dataset.

    Returns:
    DataFrame: Loaded dataset as a Pandas DataFrame.
    """
    return pd.read_csv(file_path, delimiter=r'\s+', header=None, names=[
        'area', 'perimeter', 'compactness', 'length_of_kernel',␣
 ↪'width_of_kernel', 'asymmetry_coefficient', 'length_of_kernel_groove',␣
 ↪'class'])

def explore_data(data):
    """
    Perform exploratory data analysis on the dataset.

    Parameters:
    data (DataFrame): The input data for analysis.

    Returns:
    dict: Summary statistics, missing values, and duplicate rows count.
    """
    # Summary Statistics
    summary_stats = data.describe().transpose()
    print("Summary Statistics:\n")
    print(tabulate(summary_stats, headers='keys', tablefmt='grid'))

    # Missing Values
    missing_values = data.isnull().sum().reset_index()
    missing_values.columns = ['Feature', 'Missing Values']
    print("\nMissing Values:\n")
    print(tabulate(missing_values, headers='keys', tablefmt='grid'))

    # Duplicate Rows

    duplicate_rows = pd.DataFrame({'Duplicate Rows': [data.duplicated().sum()]})
    print("\nDuplicate Rows:\n")
    print(tabulate(duplicate_rows, headers='keys', tablefmt='grid'))

    # Correlation Matrix
    print("\nCorrelation Matrix:")
```

```python
    corr_matrix = data.corr()
    plt.figure(figsize=(12, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
    plt.title('Correlation Matrix')
    plt.show()

    # Histograms
    print("\nHistograms:")
    data.hist(bins=30, figsize=(20, 15))
    plt.show()

    # Boxplots for each numeric column
    for column in data.select_dtypes(include=[np.number]).columns:
        plt.figure(figsize=(10, 6))
        sns.boxplot(x=data[column])
        plt.title(f'Boxplot of {column}')
        plt.show()

    return {"summary_stats": summary_stats, "missing_values": missing_values,␣
 ↪"duplicate_rows": duplicate_rows}


def perform_kmeans_clustering(data, n_clusters):
    """
    Perform K-means clustering on the dataset.

    Parameters:
    data (DataFrame): The input data for clustering.
    n_clusters (int): Number of clusters for K-means.

    Returns:
    DataFrame: Data with cluster labels.
    """
    # Normalize the features
    features = data.drop(columns=['class'])
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)

    # Apply K-means
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(features_scaled)

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

def plot_clusters(data, method_name):
```

```
    """
    Plot the clusters using PCA for dimensionality reduction.

    Parameters:
    data (DataFrame): The data with cluster labels.
    method_name (str): Name of the clustering method used.
    """
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(data.drop(columns=['class',␣
 ↪'cluster']))
    data_pca = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
    data_pca['cluster'] = data['cluster']

    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='PC1', y='PC2', hue='cluster', palette='viridis',␣
 ↪data=data_pca)
    plt.title(f'Clusters ({method_name})')
    plt.show()

def perform_hierarchical_clustering(data, n_clusters, linkage_type):
    """
    Perform hierarchical clustering on the dataset.

    Parameters:
    data (DataFrame): The input data for clustering.
    n_clusters (int): Number of clusters.
    linkage_type (str): Linkage type to use for clustering ('ward', 'complete',␣
 ↪'average', 'single').

    Returns:
    tuple: Data with cluster labels and silhouette score.
    """
    # Normalize the features
    features = data.drop(columns=['class'])
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)

    # Apply hierarchical clustering
    hc = AgglomerativeClustering(n_clusters=n_clusters, linkage=linkage_type)
    clusters = hc.fit_predict(features_scaled)

    # Calculate silhouette score
    score = silhouette_score(features_scaled, clusters)

    # Add cluster labels to the data
    data['cluster'] = clusters
    return data, score
```

```python
def plot_dendrogram(data, linkage_type):
    """
    Plot the dendrogram for hierarchical clustering.

    Parameters:
    data (DataFrame): The input data.
    linkage_type (str): Linkage type to use for dendrogram.
    """
    features = data.drop(columns=['class'])
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)

    linked = linkage(features_scaled, method=linkage_type)

    plt.figure(figsize=(10, 7))
    dendrogram(linked, orientation='top', distance_sort='descending',␣
 ↪show_leaf_counts=True)
    plt.title(f'Dendrogram ({linkage_type} linkage)')
    plt.show()

def perform_dbscan_clustering(data, eps, min_samples):
    """
    Perform DBSCAN clustering on the dataset.

    Parameters:
    data (DataFrame): The input data for clustering.
    eps (float): The maximum distance between two samples for them to be␣
 ↪considered as in the same neighborhood.
    min_samples (int): The number of samples in a neighborhood for a point to␣
 ↪be considered as a core point.

    Returns:
    tuple: Data with cluster labels and silhouette score.
    """
    # Normalize the features
    features = data.drop(columns=['class'])
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)

    # Apply DBSCAN
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    clusters = dbscan.fit_predict(features_scaled)

    # Calculate silhouette score
    if len(set(clusters)) > 1:  # Silhouette score is not defined for a single␣
 ↪cluster
```

```
        score = silhouette_score(features_scaled, clusters)
    else:
        score = -1  # Invalid score for a single cluster

    # Add cluster labels to the data
    data['cluster'] = clusters
    return data, score
```

[70]:
```
# Main function to run all tasks
def main():
    """
    Main function to execute all tasks:
    1. Explore the dataset.
    2. Perform K-means clustering.
    3. Perform hierarchical clustering with different linkage types.
    4. Perform DBSCAN clustering with different eps and min_samples values.
    """
    file_path = '/Users/zelalemabahana/Desktop/PennState/DAAN862/seeds_dataset.
 ↪txt'
    data = load_data(file_path)

    # Task 1: Explore the dataset
    print("Exploratory Data Analysis:\n")
    eda_summary = explore_data(data)

    # Task 2: K-means clustering
    n_clusters = 3  # Assuming there are 3 varieties of wheat
    data_kmeans = perform_kmeans_clustering(data.copy(), n_clusters)
    plot_clusters(data_kmeans, "K-means")

    # Task 3: Hierarchical clustering
    linkage_types = ['ward', 'complete', 'average', 'single']
    best_linkage_type = None
    best_score = -1

    for linkage_type in linkage_types:
        print(f"\nEvaluating linkage type: {linkage_type}")
        data_hc, score = perform_hierarchical_clustering(data.copy(),␣
 ↪n_clusters, linkage_type)
        print(f"Silhouette Score for {linkage_type} linkage: {score}")

        if score > best_score:
            best_score = score
            best_linkage_type = linkage_type

    print(f"\nBest linkage type: {best_linkage_type} with silhouette score:␣
 ↪{best_score}")
```

```
    plot_dendrogram(data, best_linkage_type)

    # Task 4: DBSCAN clustering
    eps_values = np.arange(0.1, 1.5, 0.1)
    min_samples_values = range(2, 10)

    best_eps = None
    best_min_samples = None
    best_dbscan_score = -1

    for eps in eps_values:
        for min_samples in min_samples_values:
            data_dbscan, score = perform_dbscan_clustering(data.copy(), eps,␣
↪min_samples)
            print(f"EPS: {eps}, Min Samples: {min_samples}, Silhouette Score:␣
↪{score}")

            if score > best_dbscan_score:
                best_dbscan_score = score
                best_eps = eps
                best_min_samples = min_samples

    print(f"\nBest EPS: {best_eps}, Best Min Samples: {best_min_samples}, Best␣
↪Silhouette Score: {best_dbscan_score}")
    data_dbscan_best, _ = perform_dbscan_clustering(data.copy(), best_eps,␣
↪best_min_samples)
    plot_clusters(data_dbscan_best, "DBSCAN")
```

[71]:
```
if __name__ == "__main__":
    main()
```

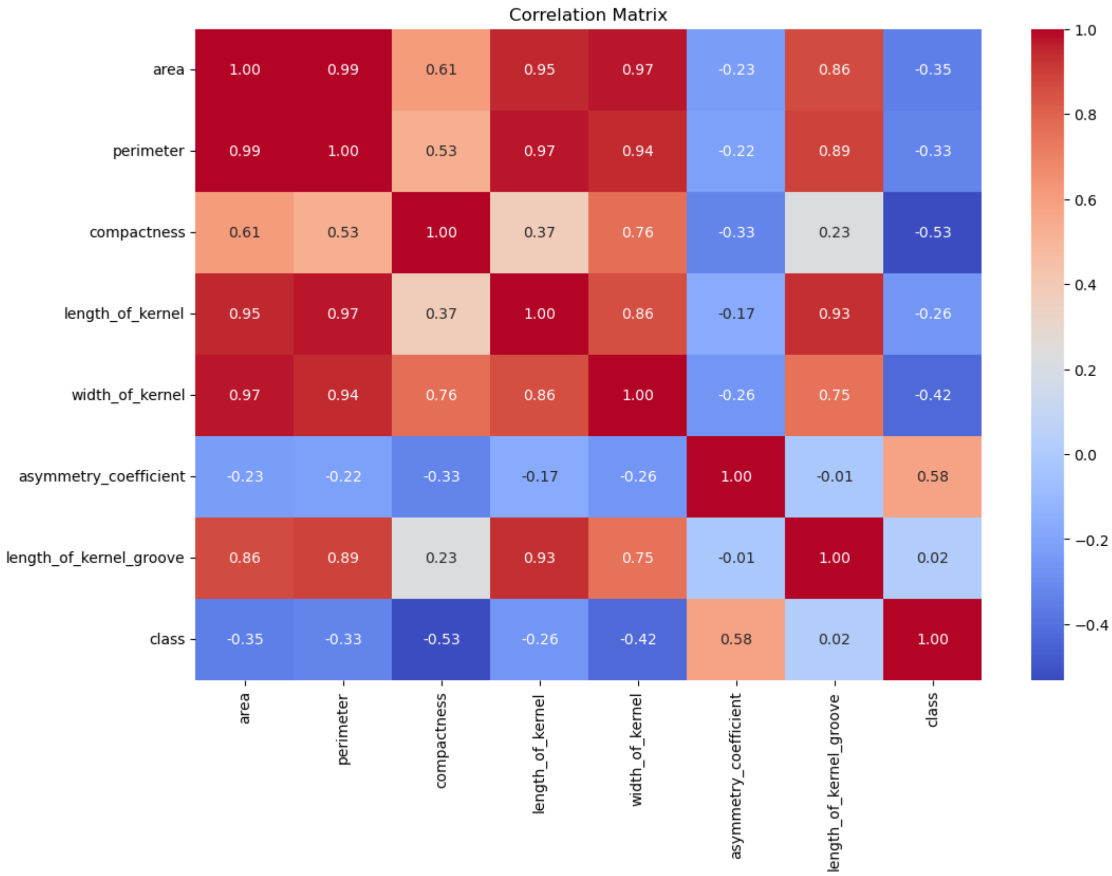
```
Exploratory Data Analysis:

Summary Statistics:

+--------------------------+---------+-----------+-----------+---------+---------
-+-----------+-----------+---------+
|                          |   count |      mean |       std |     min |      25%
|      50% |       75% |     max |
+==========================+=========+===========+===========+=========+=========
=+==========+===========+=========+
| area                     |     210 | 14.8475   | 2.9097    | 10.59   | 12.27
| 14.355   | 17.305    | 21.18   |
+--------------------------+---------+-----------+-----------+---------+---------
-+-----------+-----------+---------+
| perimeter                |     210 | 14.5593   | 1.30596   | 12.41   | 13.45
| 14.32    | 15.715    | 17.25   |
```

compactness	210	0.870999	0.0236294	0.8081	0.8569	0.87345	0.887775	0.9183
length_of_kernel	210	5.62853	0.443063	4.899	5.26225	5.5235	5.97975	6.675
width_of_kernel	210	3.2586	0.377714	2.63	2.944	3.237	3.56175	4.033
asymmetry_coefficient	210	3.7002	1.50356	0.7651	2.5615	3.599	4.76875	8.456
length_of_kernel_groove	210	5.40807	0.49148	4.519	5.045	5.223	5.877	6.55
class	210	2	0.818448	1	1	2	3	3

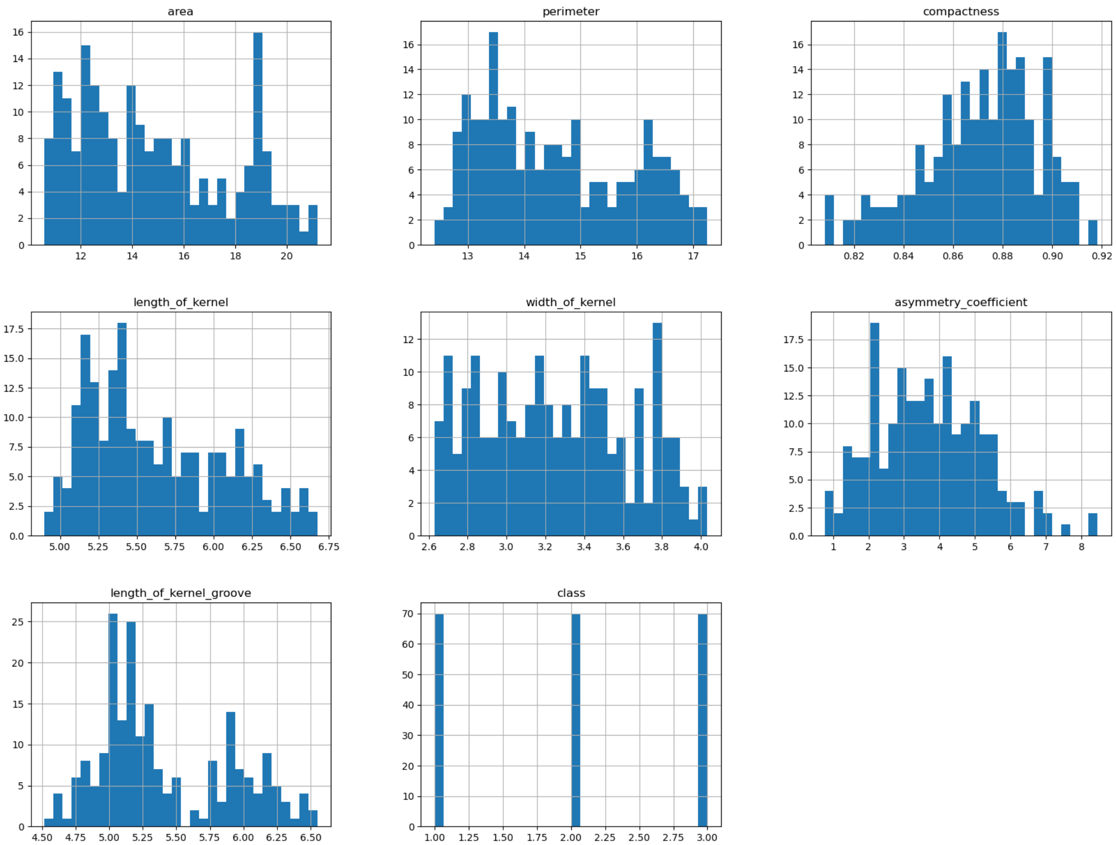
Missing Values:

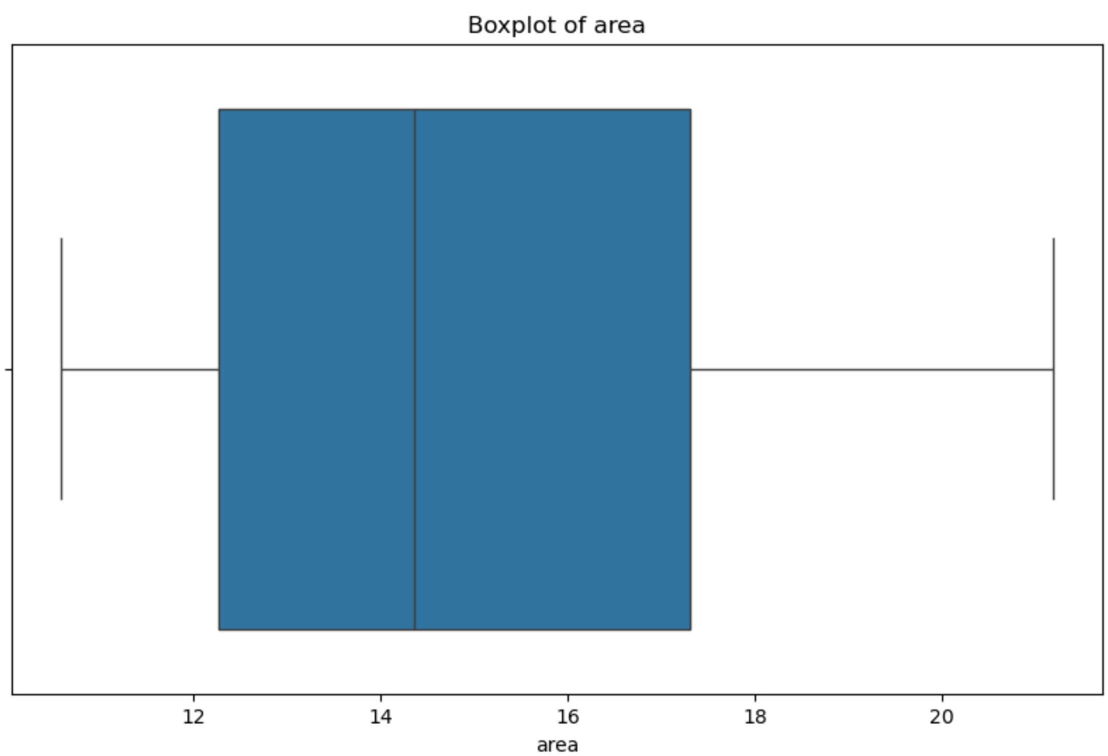
	Feature	Missing Values
0	area	0
1	perimeter	0
2	compactness	0
3	length_of_kernel	0
4	width_of_kernel	0
5	asymmetry_coefficient	0
6	length_of_kernel_groove	0
7	class	0

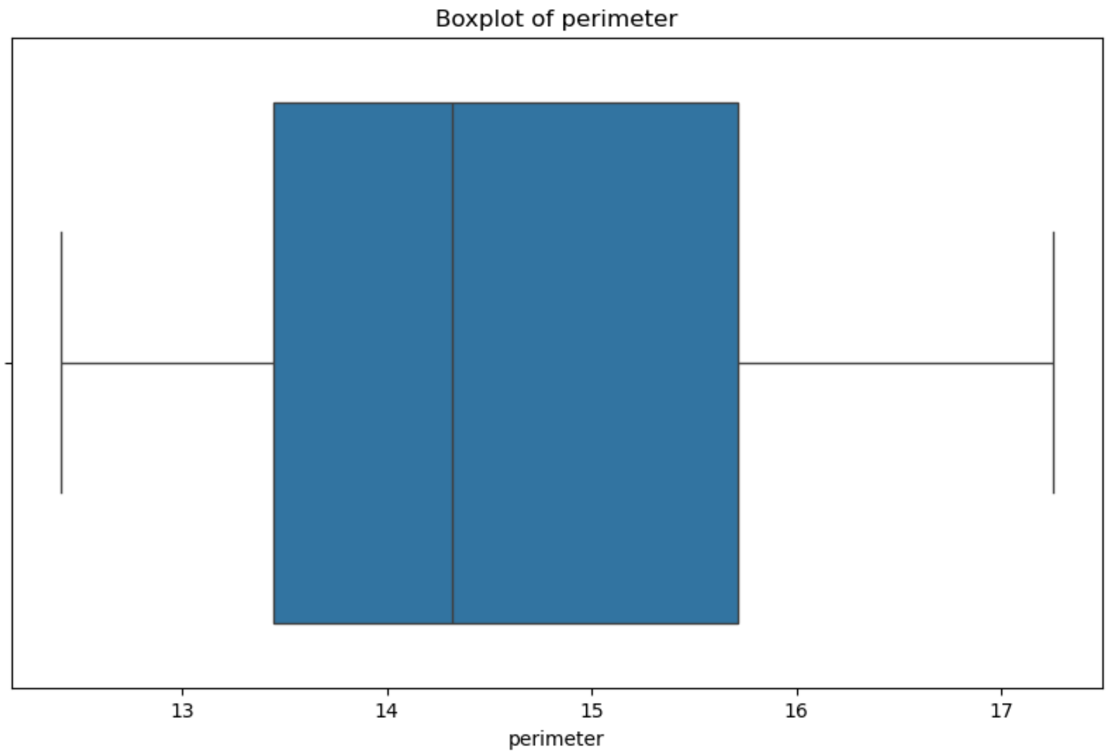
```
Duplicate Rows:

+----+------------------+
|    |   Duplicate Rows |
+====+==================+
|  0 |                0 |
+----+------------------+

Correlation Matrix:
```

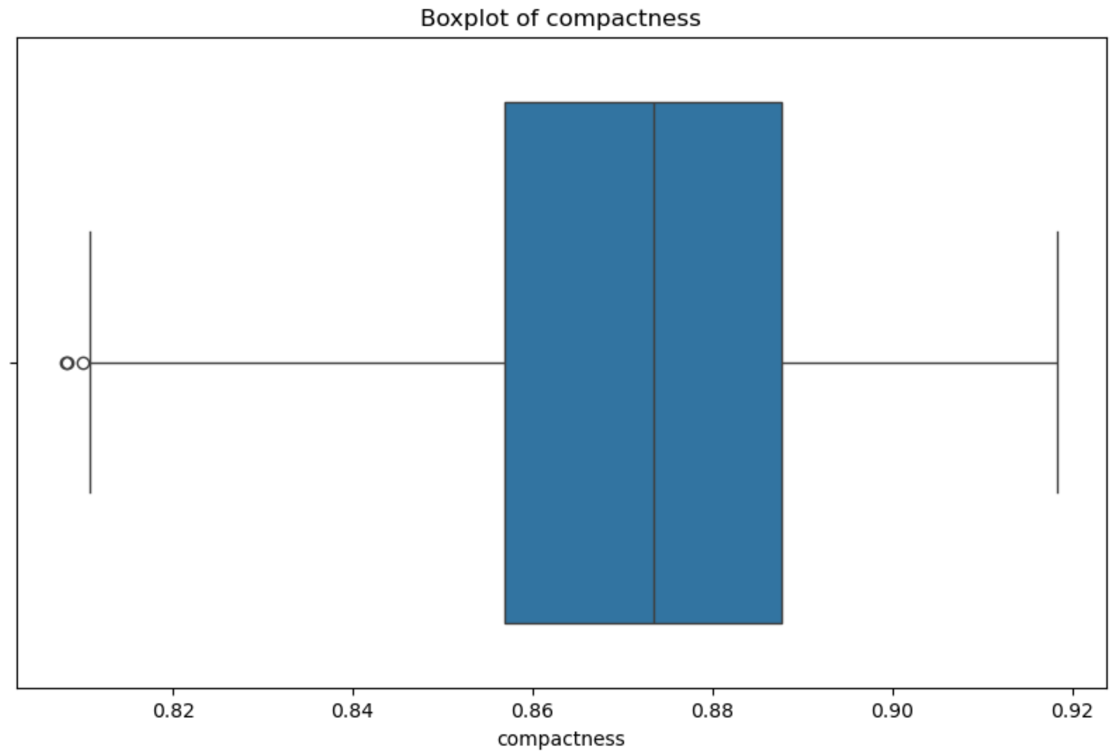


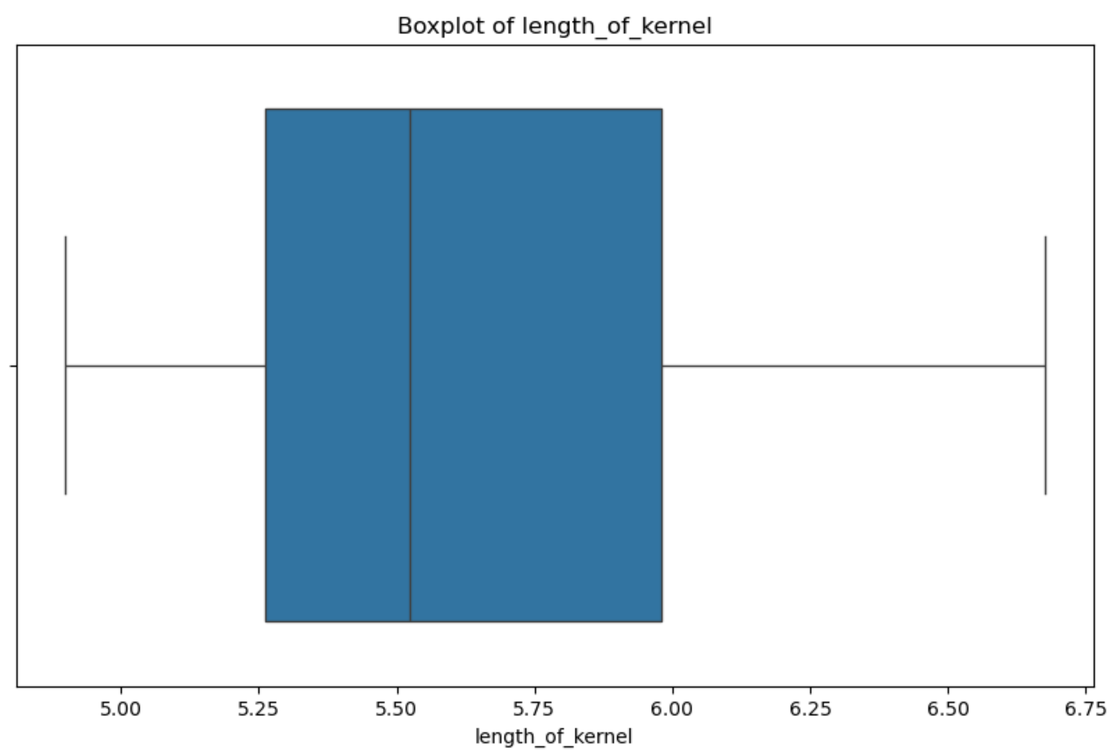

```
Histograms:
```

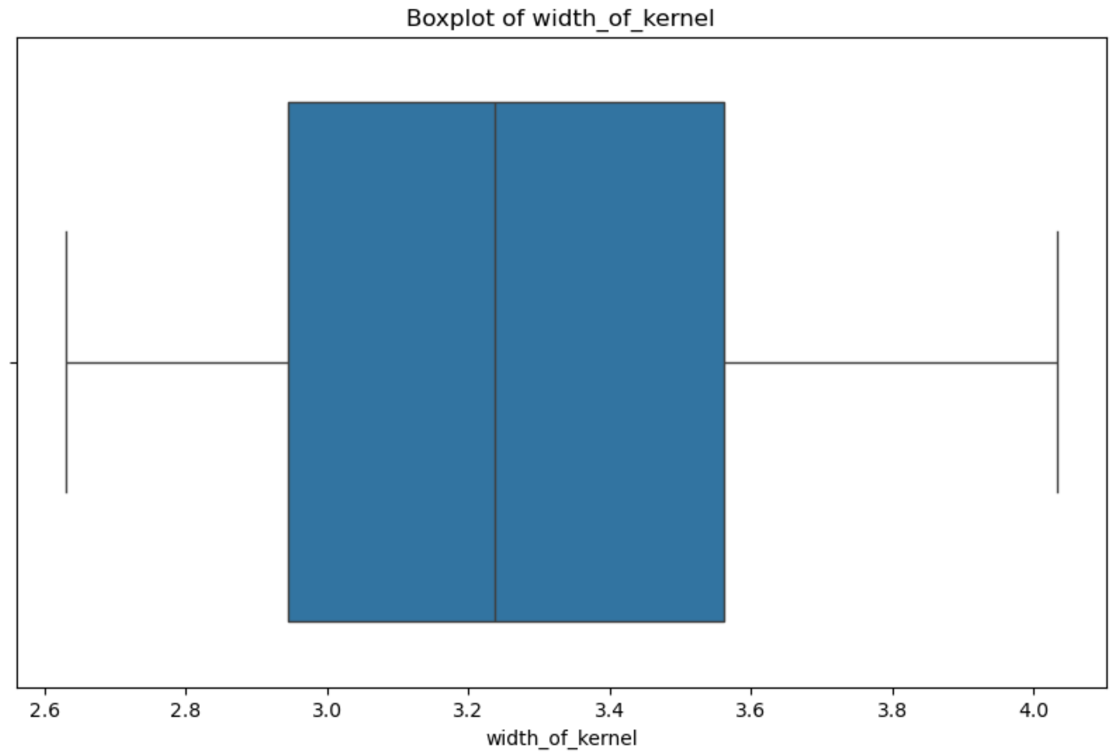

area
perimeter
compactness
length_of_kernel
width_of_kernel
asymmetry_coefficient
length_of_kernel_groove
class

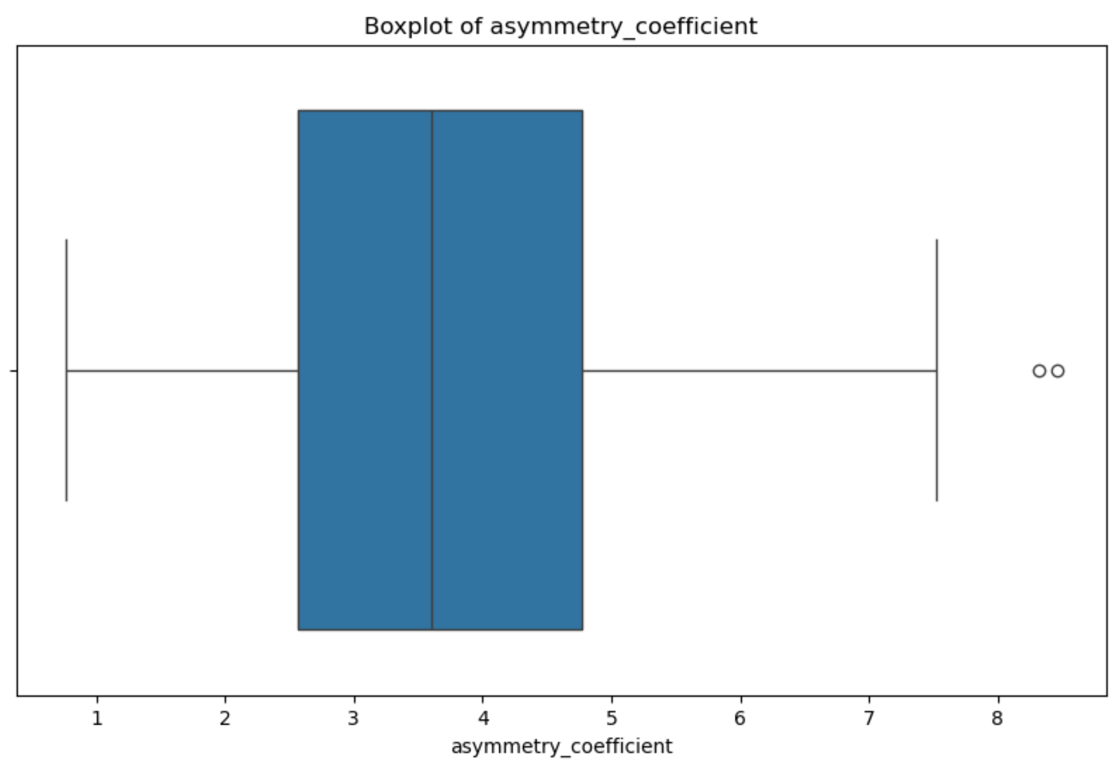

Boxplot of area
12
14
16
18
20
area

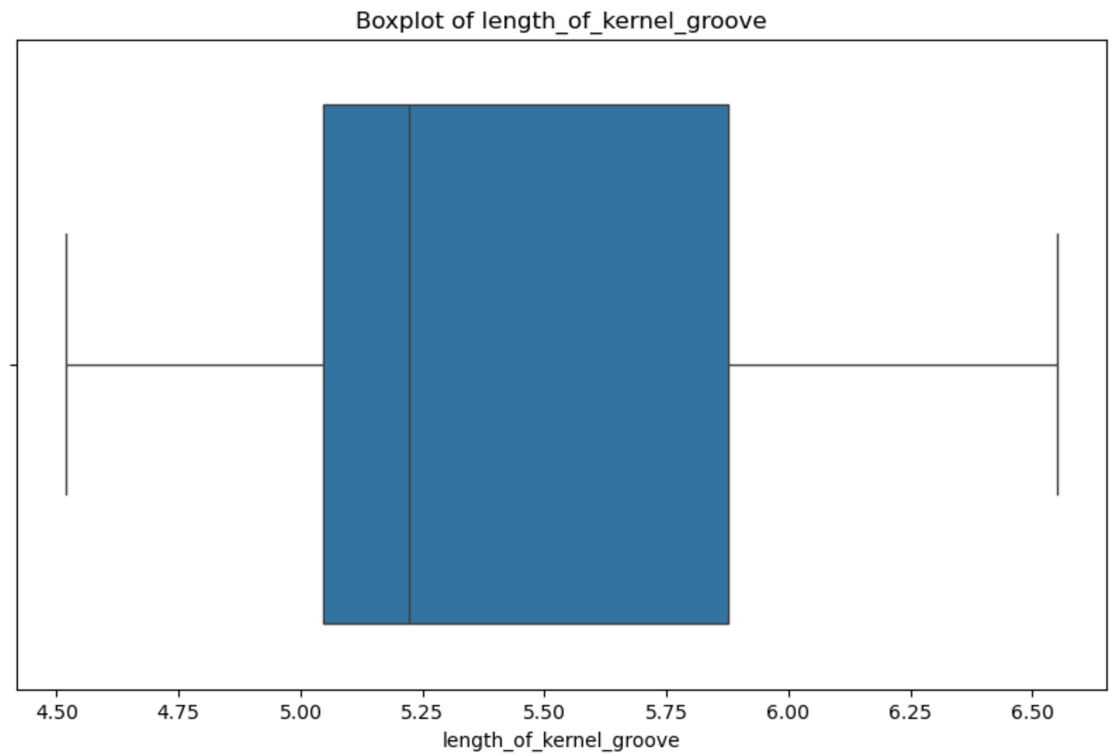

Boxplot of perimeter
13
14
15
16
17
perimeter

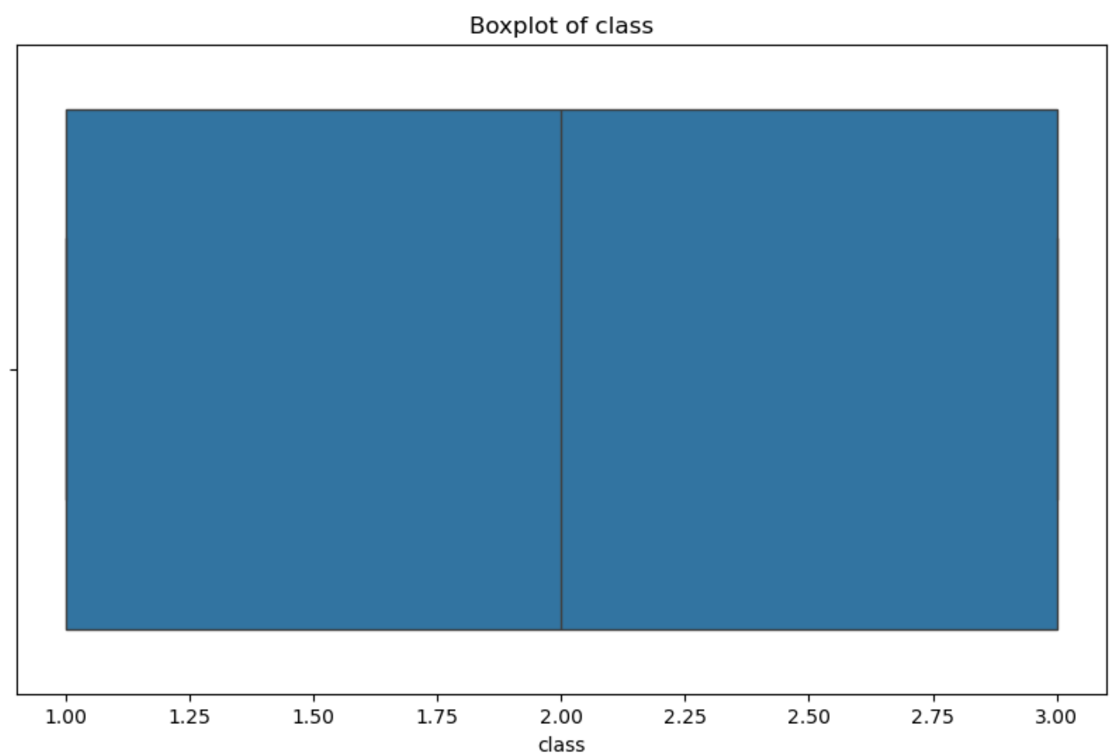

Boxplot of compactness
0.82
0.84
0.86
0.88
0.90
0.92
compactness

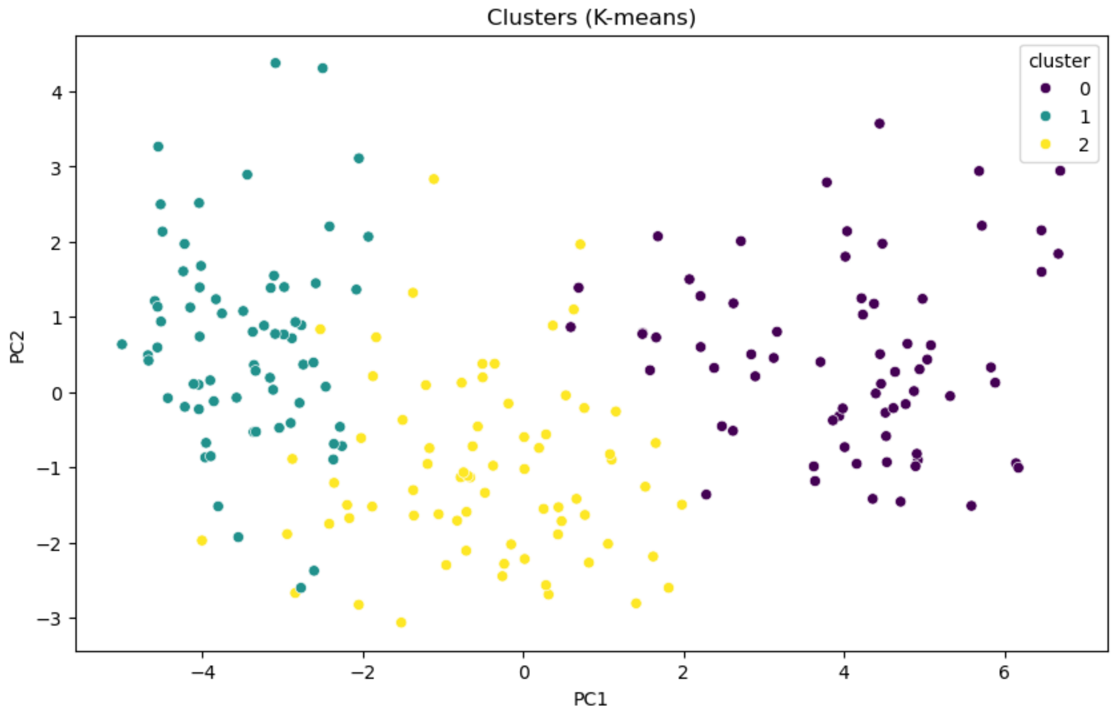

Boxplot of length_of_kernel
5.00
5.25
5.50
5.75
6.00
6.25
6.50
6.75
length_of_kernel


Boxplot of width_of_kernel
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
width_of_kernel

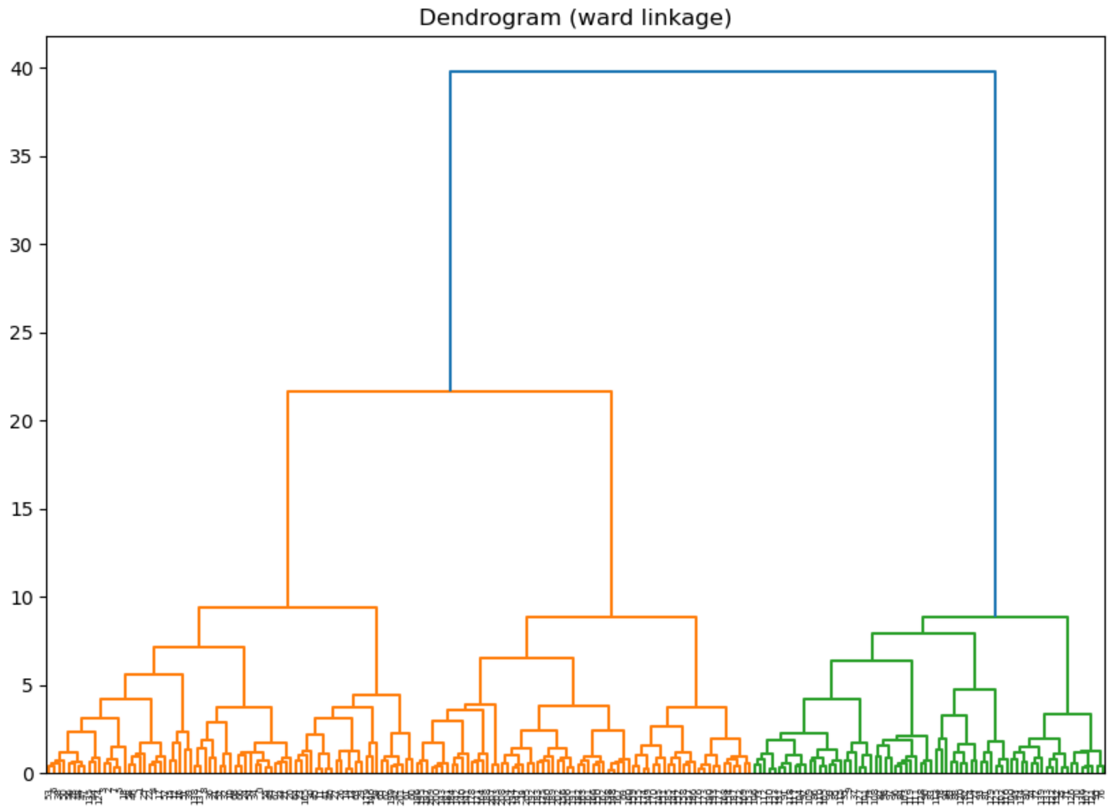

Boxplot of asymmetry_coefficient
1
2
3
4
5
6
7
8
asymmetry_coefficient


Boxplot of length_of_kernel_groove
4.50
4.75
5.00
5.25
5.50
5.75
6.00
6.25
6.50
length_of_kernel_groove


Boxplot of class
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
class



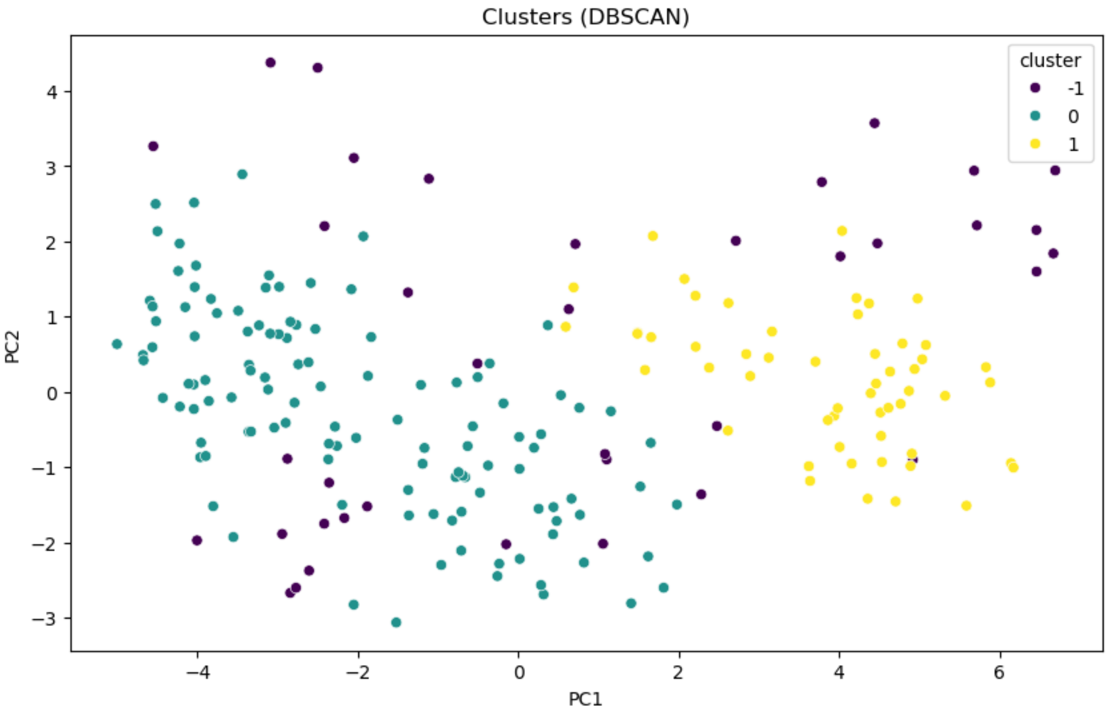

```
Evaluating linkage type: ward
Silhouette Score for ward linkage: 0.39263397091010155

Evaluating linkage type: complete
Silhouette Score for complete linkage: 0.35019845816108097

Evaluating linkage type: average
Silhouette Score for average linkage: 0.3759568059006467

Evaluating linkage type: single
Silhouette Score for single linkage: -0.005642378923309357

Best linkage type: ward with silhouette score: 0.39263397091010155
```




```
EPS: 0.1, Min Samples: 2, Silhouette Score: -1
EPS: 0.1, Min Samples: 3, Silhouette Score: -1
EPS: 0.1, Min Samples: 4, Silhouette Score: -1
EPS: 0.1, Min Samples: 5, Silhouette Score: -1
EPS: 0.1, Min Samples: 6, Silhouette Score: -1
EPS: 0.1, Min Samples: 7, Silhouette Score: -1
EPS: 0.1, Min Samples: 8, Silhouette Score: -1
EPS: 0.1, Min Samples: 9, Silhouette Score: -1
EPS: 0.2, Min Samples: 2, Silhouette Score: -0.17654483777955726
EPS: 0.2, Min Samples: 3, Silhouette Score: -1
EPS: 0.2, Min Samples: 4, Silhouette Score: -1
EPS: 0.2, Min Samples: 5, Silhouette Score: -1
EPS: 0.2, Min Samples: 6, Silhouette Score: -1
EPS: 0.2, Min Samples: 7, Silhouette Score: -1
EPS: 0.2, Min Samples: 8, Silhouette Score: -1
EPS: 0.2, Min Samples: 9, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 2, Silhouette Score: -0.4735501796872414
EPS: 0.30000000000000004, Min Samples: 3, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 4, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 5, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 6, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 7, Silhouette Score: -1
```

```
EPS: 0.30000000000000004, Min Samples: 8, Silhouette Score: -1
EPS: 0.30000000000000004, Min Samples: 9, Silhouette Score: -1
EPS: 0.4, Min Samples: 2, Silhouette Score: -0.3670313402292049
EPS: 0.4, Min Samples: 3, Silhouette Score: -0.4758040444237539
EPS: 0.4, Min Samples: 4, Silhouette Score: -1
EPS: 0.4, Min Samples: 5, Silhouette Score: -1
EPS: 0.4, Min Samples: 6, Silhouette Score: -1
EPS: 0.4, Min Samples: 7, Silhouette Score: -1
EPS: 0.4, Min Samples: 8, Silhouette Score: -1
EPS: 0.4, Min Samples: 9, Silhouette Score: -1
EPS: 0.5, Min Samples: 2, Silhouette Score: -0.18184926352510017
EPS: 0.5, Min Samples: 3, Silhouette Score: -0.3409363268767796
EPS: 0.5, Min Samples: 4, Silhouette Score: -0.4156381480384968
EPS: 0.5, Min Samples: 5, Silhouette Score: 0.06205813881401449
EPS: 0.5, Min Samples: 6, Silhouette Score: -1
EPS: 0.5, Min Samples: 7, Silhouette Score: -1
EPS: 0.5, Min Samples: 8, Silhouette Score: -1
EPS: 0.5, Min Samples: 9, Silhouette Score: -1
EPS: 0.6000000000000001, Min Samples: 2, Silhouette Score: -0.12306020541888399
EPS: 0.6000000000000001, Min Samples: 3, Silhouette Score: -0.1395546185083424
EPS: 0.6000000000000001, Min Samples: 4, Silhouette Score: -0.2326131159127796
EPS: 0.6000000000000001, Min Samples: 5, Silhouette Score: -0.29683922399450385
EPS: 0.6000000000000001, Min Samples: 6, Silhouette Score: -0.3375154672150635
EPS: 0.6000000000000001, Min Samples: 7, Silhouette Score: -0.3867575618766197
EPS: 0.6000000000000001, Min Samples: 8, Silhouette Score: -1
EPS: 0.6000000000000001, Min Samples: 9, Silhouette Score: -1
EPS: 0.7000000000000001, Min Samples: 2, Silhouette Score: -0.001966326728729347
EPS: 0.7000000000000001, Min Samples: 3, Silhouette Score: 0.014835551458973915
EPS: 0.7000000000000001, Min Samples: 4, Silhouette Score: 0.09302989214280369
EPS: 0.7000000000000001, Min Samples: 5, Silhouette Score: 0.026536009551266084
EPS: 0.7000000000000001, Min Samples: 6, Silhouette Score: -0.07518317410939843
EPS: 0.7000000000000001, Min Samples: 7, Silhouette Score: -0.13919080647670723
EPS: 0.7000000000000001, Min Samples: 8, Silhouette Score: -0.26869475308154794
EPS: 0.7000000000000001, Min Samples: 9, Silhouette Score: -0.173399418871873
EPS: 0.8, Min Samples: 2, Silhouette Score: -0.05573756813612937
EPS: 0.8, Min Samples: 3, Silhouette Score: 0.02467163034771933
EPS: 0.8, Min Samples: 4, Silhouette Score: -0.016594615697611406
EPS: 0.8, Min Samples: 5, Silhouette Score: 0.15476225627900345
EPS: 0.8, Min Samples: 6, Silhouette Score: 0.11232403470594676
EPS: 0.8, Min Samples: 7, Silhouette Score: 0.07601787657280064
EPS: 0.8, Min Samples: 8, Silhouette Score: 0.09408364520041979
EPS: 0.8, Min Samples: 9, Silhouette Score: 0.03532501637892629
EPS: 0.9, Min Samples: 2, Silhouette Score: -0.27117675230244553
EPS: 0.9, Min Samples: 3, Silhouette Score: 0.012748705388452665
EPS: 0.9, Min Samples: 4, Silhouette Score: 0.1358145169895975
EPS: 0.9, Min Samples: 5, Silhouette Score: 0.12516427403807054
EPS: 0.9, Min Samples: 6, Silhouette Score: 0.10031077735390849
EPS: 0.9, Min Samples: 7, Silhouette Score: 0.2404106044821437
```

```
EPS: 0.9, Min Samples: 8, Silhouette Score: 0.21709414004979724
EPS: 0.9, Min Samples: 9, Silhouette Score: 0.2125614967591408
EPS: 1.0, Min Samples: 2, Silhouette Score: 0.03406370133922561
EPS: 1.0, Min Samples: 3, Silhouette Score: 0.13331033993666475
EPS: 1.0, Min Samples: 4, Silhouette Score: 0.11152129872248609
EPS: 1.0, Min Samples: 5, Silhouette Score: 0.09201244928473415
EPS: 1.0, Min Samples: 6, Silhouette Score: 0.15739588129340565
EPS: 1.0, Min Samples: 7, Silhouette Score: -0.09458964007733704
EPS: 1.0, Min Samples: 8, Silhouette Score: 0.26873884854690805
EPS: 1.0, Min Samples: 9, Silhouette Score: 0.25832725420557845
EPS: 1.1, Min Samples: 2, Silhouette Score: 0.06711248789347121
EPS: 1.1, Min Samples: 3, Silhouette Score: 0.15580527834045751
EPS: 1.1, Min Samples: 4, Silhouette Score: 0.15580527834045751
EPS: 1.1, Min Samples: 5, Silhouette Score: 0.13047429996646265
EPS: 1.1, Min Samples: 6, Silhouette Score: 0.12771551380207294
EPS: 1.1, Min Samples: 7, Silhouette Score: 0.18569532291691263
EPS: 1.1, Min Samples: 8, Silhouette Score: 0.17719856336670534
EPS: 1.1, Min Samples: 9, Silhouette Score: 0.1654633699169784
EPS: 1.2000000000000002, Min Samples: 2, Silhouette Score: 0.08612318110447648
EPS: 1.2000000000000002, Min Samples: 3, Silhouette Score: 0.15921768718211454
EPS: 1.2000000000000002, Min Samples: 4, Silhouette Score: 0.15921768718211454
EPS: 1.2000000000000002, Min Samples: 5, Silhouette Score: 0.15921768718211454
EPS: 1.2000000000000002, Min Samples: 6, Silhouette Score: 0.15921768718211454
EPS: 1.2000000000000002, Min Samples: 7, Silhouette Score: 0.18014125961867528
EPS: 1.2000000000000002, Min Samples: 8, Silhouette Score: 0.15046422137818638
EPS: 1.2000000000000002, Min Samples: 9, Silhouette Score: 0.21186127976128744
EPS: 1.3, Min Samples: 2, Silhouette Score: 0.03847430740588419
EPS: 1.3, Min Samples: 3, Silhouette Score: 0.13217591755671954
EPS: 1.3, Min Samples: 4, Silhouette Score: 0.13217591755671954
EPS: 1.3, Min Samples: 5, Silhouette Score: 0.13217591755671954
EPS: 1.3, Min Samples: 6, Silhouette Score: 0.13217591755671954
EPS: 1.3, Min Samples: 7, Silhouette Score: 0.16692320974914865
EPS: 1.3, Min Samples: 8, Silhouette Score: 0.16692320974914865
EPS: 1.3, Min Samples: 9, Silhouette Score: 0.15921768718211454
EPS: 1.4000000000000001, Min Samples: 2, Silhouette Score: -1
EPS: 1.4000000000000001, Min Samples: 3, Silhouette Score: -1
EPS: 1.4000000000000001, Min Samples: 4, Silhouette Score: 0.19373291820189065
EPS: 1.4000000000000001, Min Samples: 5, Silhouette Score: 0.15045330669751378
EPS: 1.4000000000000001, Min Samples: 6, Silhouette Score: 0.15045330669751378
EPS: 1.4000000000000001, Min Samples: 7, Silhouette Score: 0.15045330669751378
EPS: 1.4000000000000001, Min Samples: 8, Silhouette Score: 0.15045330669751378
EPS: 1.4000000000000001, Min Samples: 9, Silhouette Score: 0.15045330669751378

Best EPS: 1.0, Best Min Samples: 8, Best Silhouette Score: 0.26873884854690805
```



Task 1. Explore the data set:

Summary Statistics and Observations:

- The dataset consists of 210 instances with 8 attributes.
- The dataset includes various measurements, such as area, perimeter, compactness, length_of_kernel, width_of_kernel, asymmetry_coefficient, length_of_kernel_groove and class.
- There are no missing values and duplicates in our dataset
- Area, perimeter, and length of kernel show a more left-skewed distribution, and compactness appear to be right skewed. Length of kernel groove appears to have a distribution more heaviter on the tails than centered.
- There are three classes in the class distribution.
- Strong correlations can be observed between asymmetry coefficient, compactness, and width of kernel.
- Feature Transformation: features are normalized using 'standard sclaler'.

Task 2: Use K-means clustering to group the seed data: - K-means clustering was applied to group the seed data, and three groups we showed in distinct colours.

Task 3: Use different linkage type for Hierarchical clustering to the seed data, which linkage type give the best result:

Linkage types:

1. 'Ward' method minimizes the variance of the clusters being merged. It aims to create clusters that are as homogeneous as possible.

2. 'Complete' linkage considers the maximum distance between points in different clusters.
3. 'Average' linkage calculates the average distance between points in different clusters.
4. 'Single' linkage considers the minimum distance between points in different clusters.

- Silhouette Score a measures how similar an object is to its own cluster compared to other clusters. It combines both cohesion and separation.
- I chose linkage types 'ward', 'complete', 'average' and 'single', and compared their performance using Silhouette Score. The best linkage type is 'ward' with the highest Silhouette Score of .39263.

Evaluating linkage type: - Ward: 0.39263 - Complete: 0.35019 - Average: 0.37595 - Single: -0.00564

Task 4: Use DBscan clustering group the seed data and find the best epses and min_samples value:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that is particularly well-suited for identifying clusters in datasets with noise and outliers.

Best clustering with Silhouette Score of 0.268738 has epses values of 1.0, and Min Sample of 8.

Summary of Findings:

- The dataset and features we assessed using statistical summaries, disributions, check on outliers, and correlations. For the clustering exercise, the features were normalized and best choosen clusters were visualized.
- The silhouette score is calculated to compare the performance of different linkage types for hierarchical clustering, and different eps and min_samples values were considered for DBSCAN.
- Four linkage types and their effects on clustering are measured. Based on silhouette score, the best-performing clustering method is using the 'ward' linkage method.
- It appears that clustering using the 'ward' method performed better than DBscan clustering which suggests that the clusters formed by the 'ward' method were more well-defined and distinct from each other.

[ ]:
```

```