DAT565/DIT407 Assignment 5

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-02-23

This paper is addressing the assignment 3 study queries within the *Introduction to Data Science & AI* course, DIT407 at the University of Gothenburg and DAT565 at Chalmers. The main source of information for this project is derived from the lectures and Skiena [1]. Assignment 5 is about distance and network methods.

Problem 1: Preprocessing the dataset

To preprocess the dataset, we first load the data from the file `seeds.csv` using the `pandas` library. We then normalize the data using the `StandardScaler` from `sklearn`.

Problem 2: Determining the appropriate number of clusters

The normalized data is then used to calculate the inertia for different numbers of clusters. The inertia is plotted in Figure 1. Using the elbow method, we can see that the optimal number of clusters is 3.

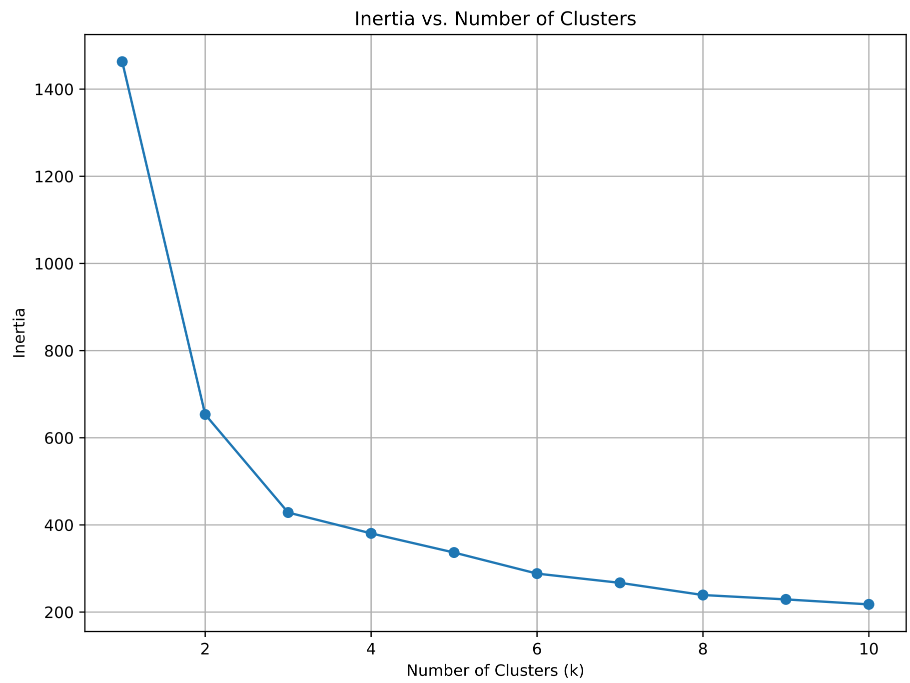

Figure 1: Invertia vs. Number of clusters

Problem 3: Visualizing the classes

To visually represent the classes, we initially project all the features onto a 2D plane. It becomes evident that the seeds form distinct clusters corresponding to the three species groups. However, certain features exhibit less distinct separation than others, such as asymmetry and compactness. Notably, the scatter plot depicting the relationship between seed length and width clearly illustrates that seed sizes are readily discernible. This is illustrated in Figure 2.

Subsequently, we employ Gaussian random projection to condense the data into a 2D space. The outcome is depicted in Figure 3, where the different seed species remain distinguishable. Finally, the UMAP projection is presented in Figure 4, highlighting the distinctiveness of the seed species.

The seeds are indeed distinguishable in the 2D space, particularly through size-related features (area, perimeter, length, width), which exhibit a linear relationship. Conversely, other features demonstrate less separability

The ability to distinguish seeds based on size-related features in a 2D space implies that clustering algorithms, particularly those sensitive to geometric relationships, can effectively separate seed samples into distinct groups. Algorithms like K-means or hierarchical clustering can leverage these separable features to identify natural clusters within the data. However, features that are less separable might pose challenges for clustering algorithms, potentially leading to overlapping clusters.

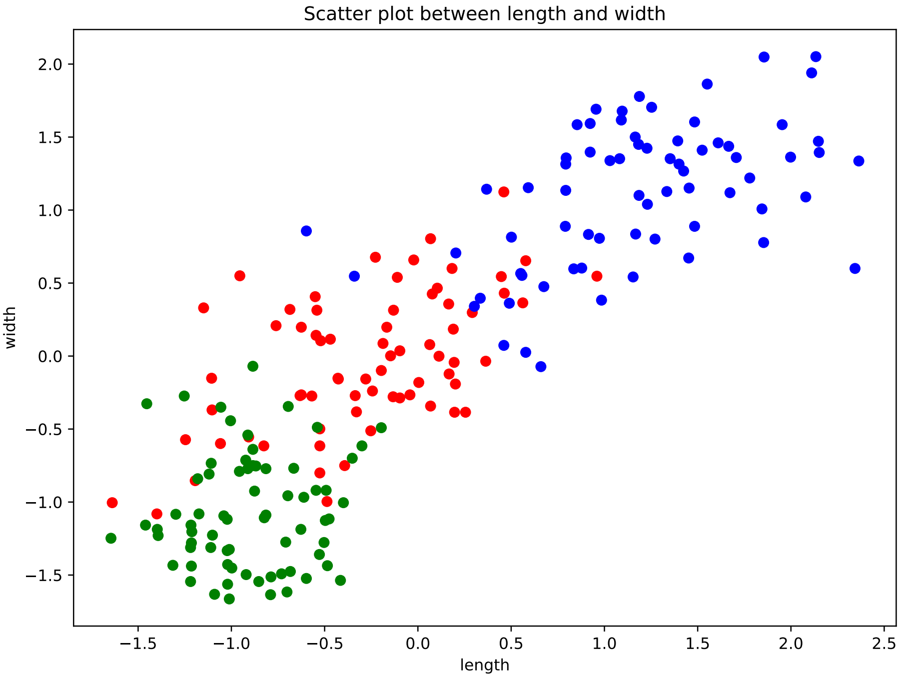

Figure 2: Scatter plot between lenght and width

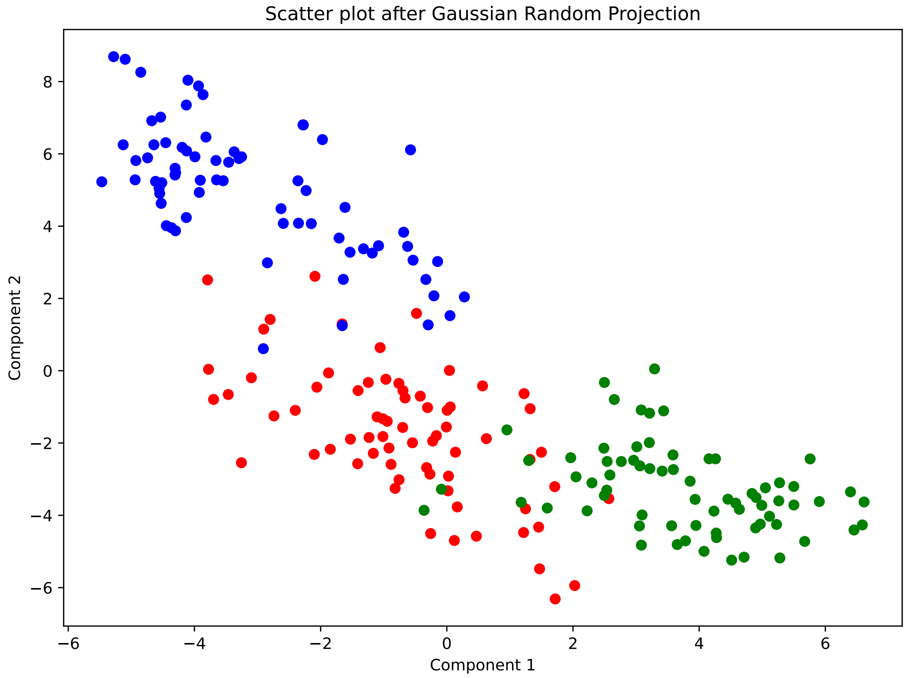

Figure 3: Gaussian random projection

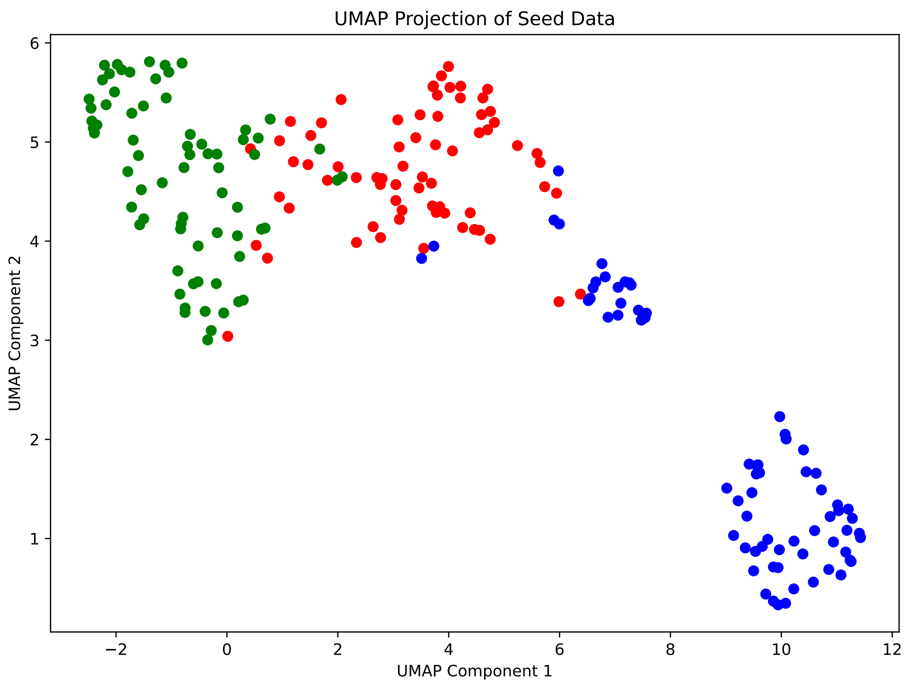

Figure 4: UMAP projection of Seeds

Problem 4: Evaluating clustering

To apply k-means clustering to the data, we use the `KMeans` function from `sklearn` with 3 as the number of clusters, and then build the model on the normalized data.

The rand index is obtained by applying the `rand_score` function on the labels of the clustering and the true labels. Its value is 0.90.

Finally we iterate over all the possible permutations in the range $[0..4]$ to find the best accuracy score. With the permutation $\{0, 1, 2, 3\} \rightarrow \{2, 3, 1, 0\}$, the accuracy is equal to 0.92.

Problem 5: Agglomerative clustering

We iterate over the linkage options and calculate the accuracy value after finding the right permutation for each of the linkage options. The best linkage option is the ward method, with an accuracy of 0.93. The dendrogram is shown in Figure 5

By looking at the 2-dimension projections from Problem 3, some of the points are close neighbors to points that don't belong to the same cluster, and the boundaries between clusters are not clearly defined. Therefore the "single" linkage option which merge clusters depending on the minimum distance gives a low accuracy value of 0.35. Other linkage options give roughly the same accuracy (around 0.9).

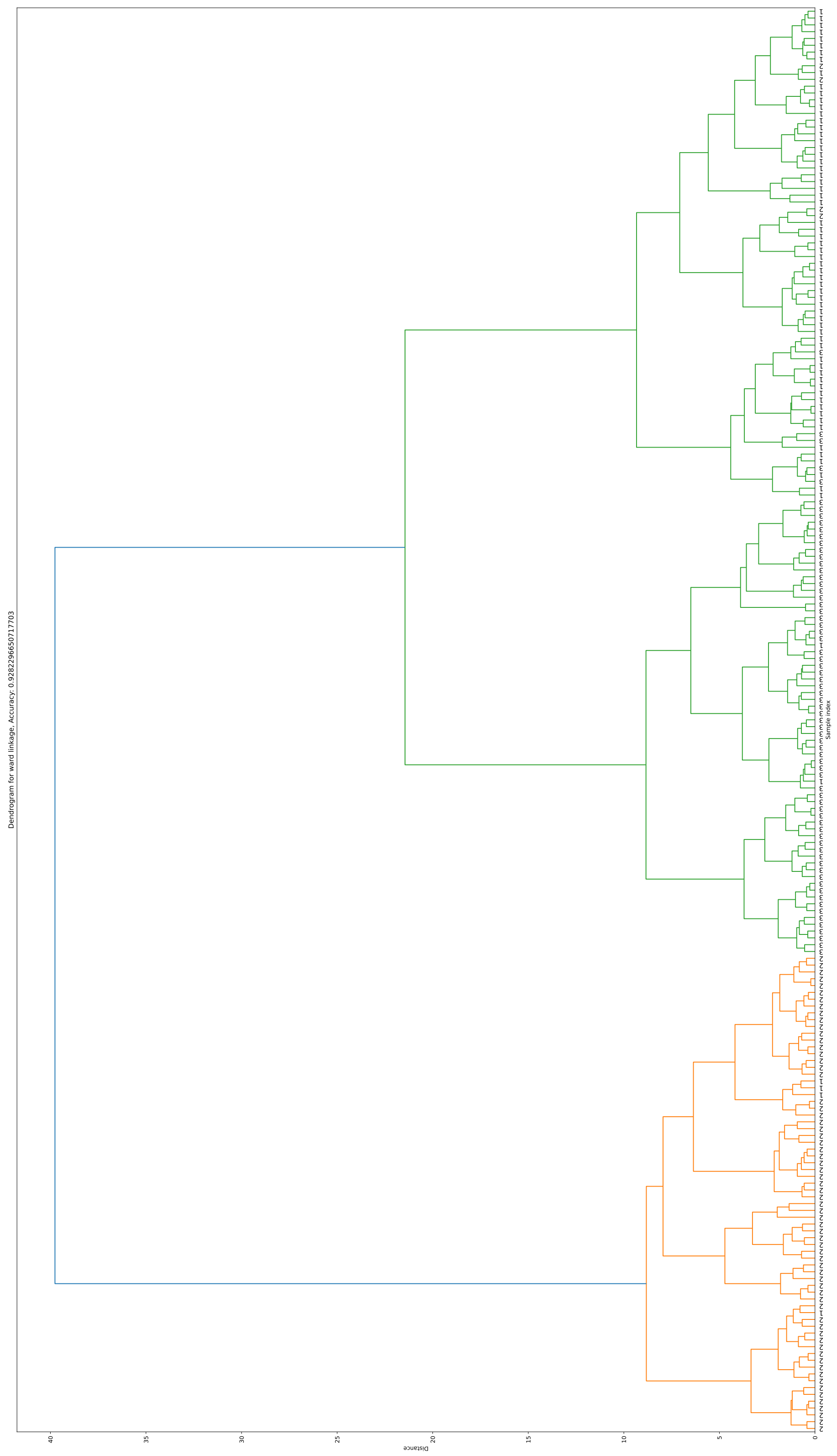

Figure 5: Dendrogram

References

[1] Steven S Skiena. *The Data Science Design Manual*. Retrieved 2024-01-20. 2024. URL: https://ebookcentral.proquest.com/lib/gu/detail.action?docID=6312797.

Appendix: Source Code

```python
from umap import UMAP
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.random_projection import GaussianRandomProjection
from sklearn.metrics import rand_score
import itertools
from sklearn.metrics import accuracy_score
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

# Load the seeds dataset
random_state = 79
seeds = pd.read_table('Assignment5/seeds.tsv')
seeds.columns = ['area', 'perimeter', 'compactness', 'length', 'width', 'asymmetry', 'groove', 'species']

X = seeds.drop(columns=['species'])  # Features
y = seeds['species']

# Normalize the data
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

seeds_normalized = pd.DataFrame(X_normalized, columns=X.columns)
seeds_normalized['species'] = y

X = seeds_normalized.drop(columns=['species'])

def plot_inertia(X):
    inertia_values = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=random_state).fit(X)
        inertia_values.append(kmeans.inertia_)

    plt.plot(range(1, 11), inertia_values, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia')
    plt.title('Inertia vs. Number of Clusters')
    plt.grid(True)
    plt.show()

def plot_features(features, y, colors):
    num_features = len(features)
    num_rows = num_features - 1
    num_cols = num_features - 1

    fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 15))

    for i in range(num_rows):
```

```python
        for j in range(num_cols):
            if i != j:
                ax = axes[i, j]
                ax.scatter(X[features[i]], X[features[j]], c=y.map(colors))
                ax.set_xlabel(features[i])
                ax.set_ylabel(features[j])
                ax.set_title(f'Scatter plot between {features[i]} and {features[j]}')

    plt.tight_layout()
    plt.show()

def plot_gaussian_random_projection(X, y, colors):
    grp = GaussianRandomProjection(n_components=2, random_state=random_state)
    projected = grp.fit_transform(X)

    plt.figure(figsize=(8, 6))
    plt.scatter(projected[:, 0], projected[:, 1], c=y.map(colors))
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.title('Scatter plot after Gaussian Random Projection')
    plt.show()

def plot_umap(X, y, colors):
    umap_model = UMAP(n_components=2)
    umap = umap_model.fit_transform(X)

    plt.figure(figsize=(8, 6))
    plt.scatter(umap[:, 0], umap[:, 1], c=y.map(colors))
    plt.xlabel('UMAP Component 1')
    plt.ylabel('UMAP Component 2')
    plt.title('UMAP Projection of Seed Data')
    plt.show()



def find_permutation(n_clusters, true_labels, cluster_labels):
    permutations = itertools.permutations(range(n_clusters))
    best_permutation = None
    best_accuracy = 0
    for permutation in permutations:
        permuted_labels = [permutation[label] for label in cluster_labels]
        accuracy = accuracy_score(permuted_labels, true_labels)
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_permutation = permutation
    return best_permutation, best_accuracy


def plot_dendrogram(n_clusters, X, y):
    linkage_options = ['ward', 'complete', 'average', 'single']
    best_accuracy = 0
    best_linkage = None

    for linkage_option in linkage_options:
        clustering = AgglomerativeClustering(n_clusters=len(y.unique()), linkage=linkage_option)
        cluster = clustering.fit(X)
        permutation, accuracy = find_permutation(n_clusters, y,
```

```
cluster.labels_)

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_linkage = linkage_option

    Z = linkage(X, method=best_linkage)
    plt.figure(figsize=(12, 6))
    dendrogram(Z, labels=y.values, leaf_rotation=90, leaf_font_size=8)
    plt.title(f"Dendrogram for {best_linkage} linkage, Accuracy: {best_accuracy}")
    plt.xlabel("Sample index")
    plt.ylabel("Distance")
    plt.show()

plot_inertia(X)
colors = {1: 'red', 2: 'blue', 3: 'green'}
features = seeds_normalized.columns
plot_features(features, y, colors)
plot_gaussian_random_projection(X, y, colors)
plot_umap(X, y, colors)


kmeans = KMeans(n_clusters=len(y.unique()), random_state=random_state)
kmeans.fit(X)
kmeans_labels = kmeans.labels_

rand_index = rand_score(y, kmeans_labels)
print("Rand score:", rand_index)

all_labels = pd.Series(kmeans_labels)._append(y)
all_unique_labels = all_labels.unique()

best_permutation, best_accuracy = find_permutation(len(all_unique_labels), y, kmeans_labels)

print("Best Accuracy:", best_accuracy)
print("Best Permutation:", best_permutation)

plot_dendrogram(len(all_unique_labels), X, y)
```