

# DAT565/DIT407 Assignment 5

Ola Bratt  
ola.bratt@gmail.com

Patrick Attimont  
patrickattimont@gmail.com

2024-02-xx

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.

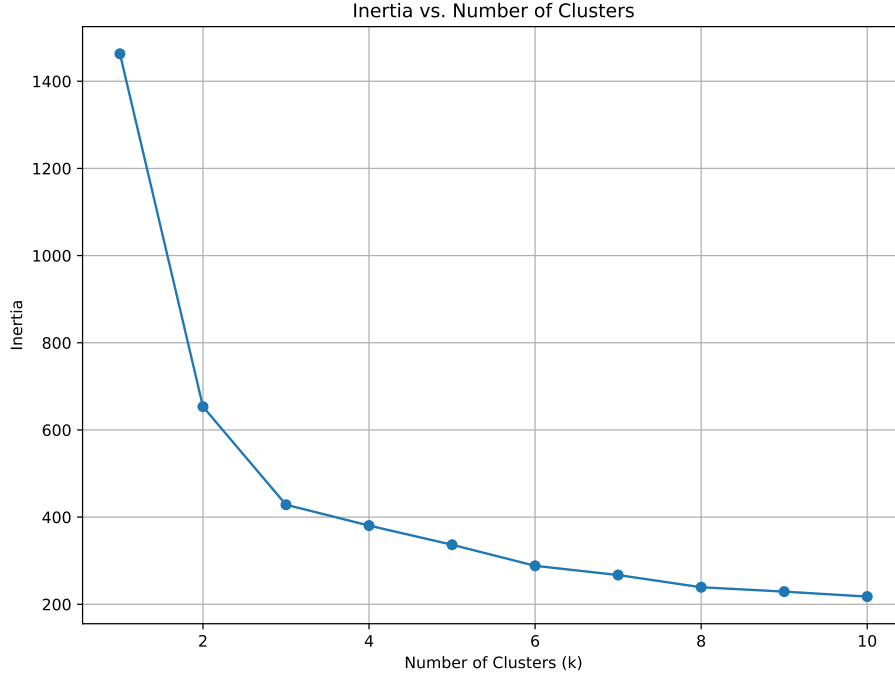


Figure 1: Inertia vs. Number of clusters

### 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.

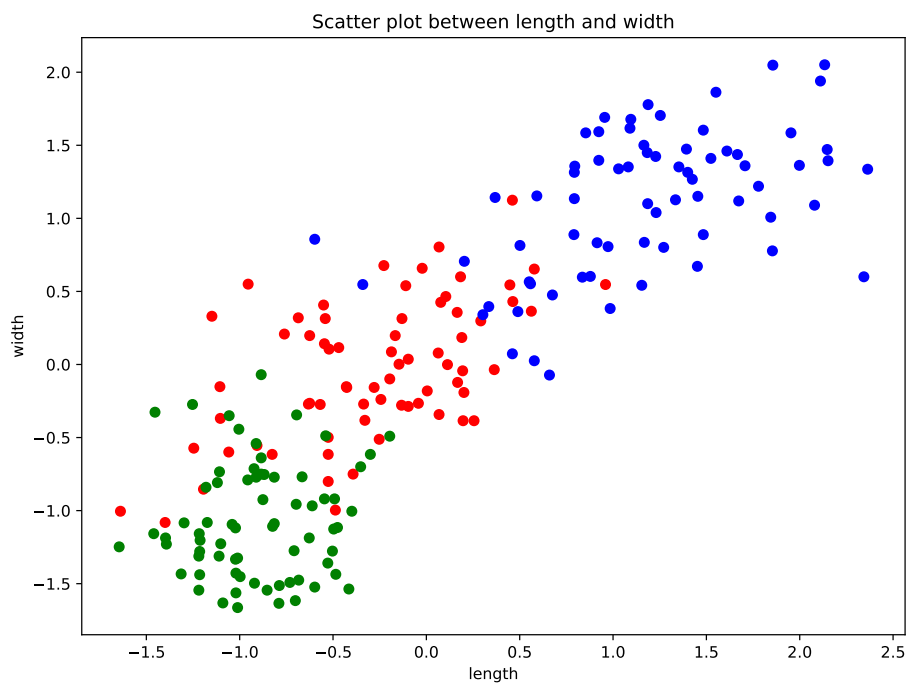


Figure 2: Scatter plot between lenght and width

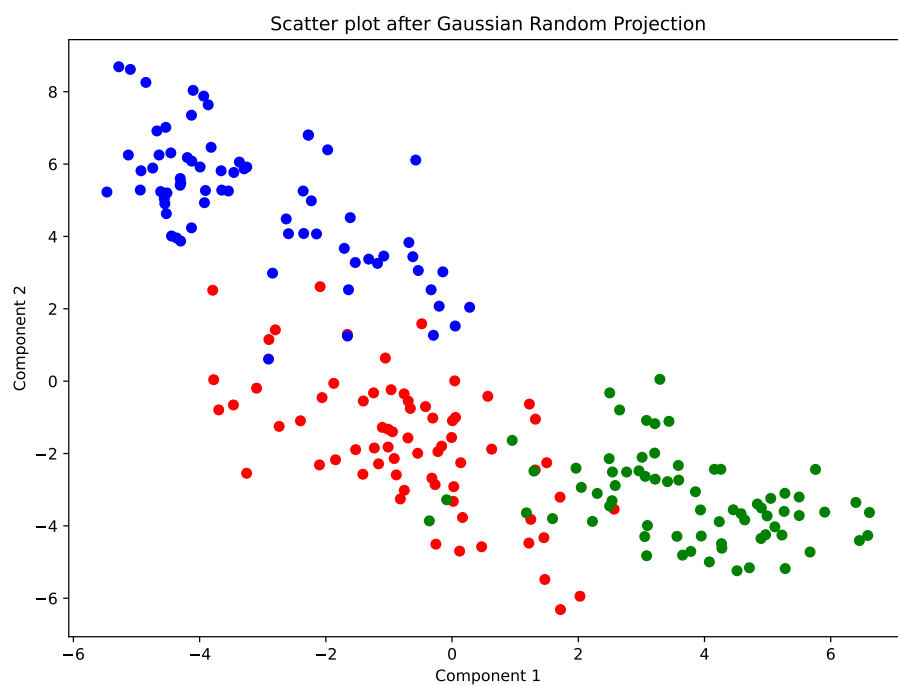


Figure 3: Gaussian random projection

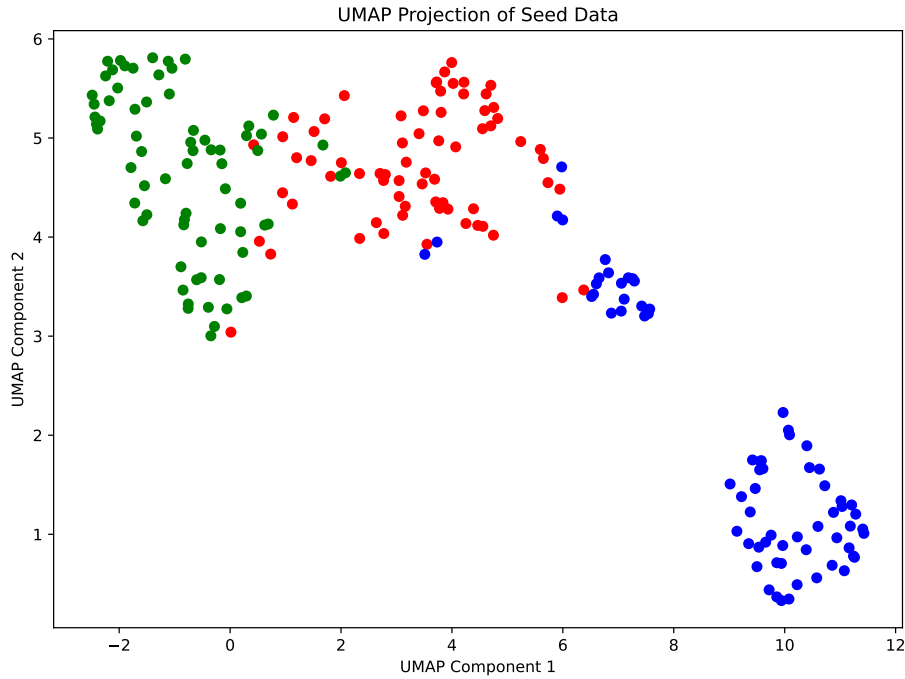


Figure 4: UMAP projection of Seeds

## 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).

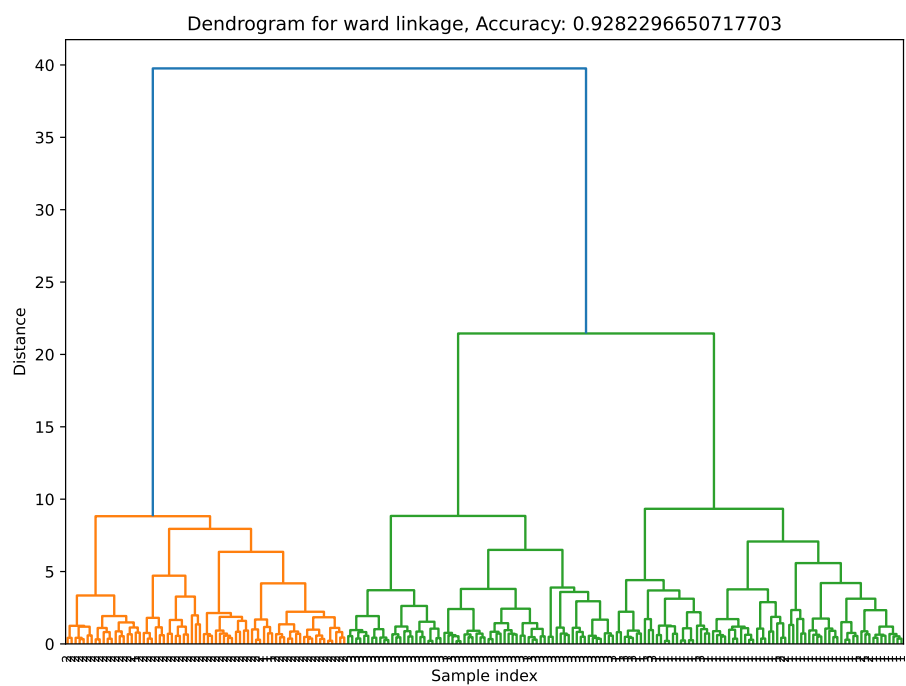


Figure 5: Dendrogram

## 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

```
1 from umap import UMAP
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.cluster import KMeans
6 from sklearn.random_projection import GaussianRandomProjection
7 from sklearn.metrics import rand_score
8 import itertools
9 from sklearn.metrics import accuracy_score
10 from scipy.cluster.hierarchy import dendrogram, linkage
11 from sklearn.cluster import AgglomerativeClustering
12
13 # Load the seeds dataset
14 random_state = 79
15 seeds = pd.read_table('Assignment5/seeds.tsv')
16 seeds.columns = ['area', 'perimeter', 'compactness', 'length', '
    ↳ width', 'asymmetry', 'groove', 'species']
17
18 X = seeds.drop(columns=['species']) # Features
19 y = seeds['species']
20
21 # Normalize the data
22 scaler = StandardScaler()
23 X_normalized = scaler.fit_transform(X)
24
25 seeds_normalized = pd.DataFrame(X_normalized, columns=X.columns)
26 seeds_normalized['species'] = y
27
28 X = seeds_normalized.drop(columns=['species'])
29
30 def plot_inertia(X):
31     inertia_values = []
32     for k in range(1, 11):
33         kmeans = KMeans(n_clusters=k, random_state=random_state).
34             ↳ fit(X)
35         inertia_values.append(kmeans.inertia_)
36
37     plt.plot(range(1, 11), inertia_values, marker='o')
38     plt.xlabel('Number-of-Clusters-(k)')
39     plt.ylabel('Inertia')
40     plt.title('Inertia-vs.-Number-of-Clusters')
41     plt.grid(True)
42     plt.show()
43
44 def plot_features(features, y, colors):
45     num_features = len(features)
46     num_rows = num_features - 1
47     num_cols = num_features - 1
48
49     fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 15))
50
51     for i in range(num_rows):
```

```

51         for j in range(num_cols):
52             if i != j:
53                 ax = axes[i, j]
54                 ax.scatter(X[features[i]], X[features[j]], c=y.map(
55                     ↪ colors))
56                 ax.set_xlabel(features[i])
57                 ax.set_ylabel(features[j])
58                 ax.set_title(f'Scatter plot between-{features[i]}-
59                     ↪ and-{features[j]}')
60
61     plt.tight_layout()
62     plt.show()
63
64 def plot_gaussian_random_projection(X, y, colors):
65     grp = GaussianRandomProjection(n_components=2, random_state=
66         ↪ random_state)
67     projected = grp.fit_transform(X)
68
69     plt.figure(figsize=(8, 6))
70     plt.scatter(projected[:, 0], projected[:, 1], c=y.map(colors))
71     plt.xlabel('Component-1')
72     plt.ylabel('Component-2')
73     plt.title('Scatter plot after Gaussian Random Projection')
74     plt.show()
75
76 def plot_umap(X, y, colors):
77     umap_model = UMAP(n_components=2)
78     umap = umap_model.fit_transform(X)
79
80     plt.figure(figsize=(8, 6))
81     plt.scatter(umap[:, 0], umap[:, 1], c=y.map(colors))
82     plt.xlabel('UMAP-Component-1')
83     plt.ylabel('UMAP-Component-2')
84     plt.title('UMAP Projection of Seed Data')
85     plt.show()
86
87 def find_permutation(n_clusters, true_labels, cluster_labels):
88     permutations = itertools.permutations(range(n_clusters))
89     best_permutation = None
90     best_accuracy = 0
91     for permutation in permutations:
92         permuted_labels = [permutation[label] for label in
93             ↪ cluster_labels]
94         accuracy = accuracy_score(permuted_labels, true_labels)
95         if accuracy > best_accuracy:
96             best_accuracy = accuracy
97             best_permutation = permutation
98     return best_permutation, best_accuracy
99
100 def plot_dendrogram(n_clusters, X, y):
101     linkage_options = ['ward', 'complete', 'average', 'single']
102     best_accuracy = 0
103     best_linkage = None
104
105     for linkage_option in linkage_options:
106         clustering = AgglomerativeClustering(n_clusters=len(y),
107             ↪ unique()), linkage=linkage_option)
108         cluster = clustering.fit(X)
109         permutation, accuracy = find_permutation(n_clusters, y,

```



```

108         ↪ cluster.labels_)
109     if accuracy > best_accuracy:
110         best_accuracy = accuracy
111         best_linkage = linkage_option
112
113     Z = linkage(X, method=best_linkage)
114     plt.figure(figsize=(12, 6))
115     dendrogram(Z, labels=y.values, leaf_rotation=90, leaf_font_size
116         ↪ =8)
117     plt.title(f"Dendrogram for {best_linkage} linkage, Accuracy: {
118         ↪ best_accuracy}")
119     plt.xlabel("Sample index")
120     plt.ylabel("Distance")
121     plt.show()
122
123     plot_inertia(X)
124     colors = {1: 'red', 2: 'blue', 3: 'green'}
125     features = seeds.normalized.columns
126     plot_features(features, y, colors)
127     plot_gaussian_random_projection(X, y, colors)
128     plot_umap(X, y, colors)
129
130     kmeans = KMeans(n_clusters=len(y.unique()), random_state=
131         ↪ random_state)
132     kmeans.fit(X)
133     kmeans.labels = kmeans.labels_
134
135     rand_index = rand_score(y, kmeans.labels)
136     print("Rand-score:", rand_index)
137
138     all_labels = pd.Series(kmeans.labels).append(y)
139     all_unique_labels = all_labels.unique()
140
141     best_permutation, best_accuracy = find_permutation(len(
142         ↪ all_unique_labels), y, kmeans.labels)
143
144     print("Best Accuracy:", best_accuracy)
145     print("Best Permutation:", best_permutation)
146
147     plot_dendrogram(len(all_unique_labels), X, y)

```