DIT407 Introduction to data science and AI
Assignment 5

Reynir Siik
reynir.siik@gmail.com

Franco Zambon
guszamfr@student.gu.se

2024-02-25

Abstract

This text is from the file `abstract.tex` and will be included in the abstract.

1 Problem 1: Preprocessing the dataset

The data in the file have different metrics and units. To be able to do a meaningful clustering the data needs to be normalized. A method that distorts the shape of the data as little as possible is the Z-score method. The z-score method is chosen on those premises.

2 Problem 2: Determining the appropriate number of clusters

With k-means clustering and calculation of inertia the elbow method is used to find the number of clusters.1 While there is a big difference between 1-2 and 2-3 clusters, there are almost equal distance between the following clusters. The number of clusters the model predicts seems to be three.

When applying linear regression to the "lower arm" part of the elbow diagram and calculating Mean Squared Error (MSE) for 2, 3 and 4 clusters the following approximate numbers will be the results:

When looking at the table a clear elbow effect can be observed when going from 3 to 4 clusters. This would support the presumption that there is three clusters.

Clusters	MSE
2	30 600
3	9 500
4	6 400
5	3 500

Table 1: MSE resulting from various number of clusters

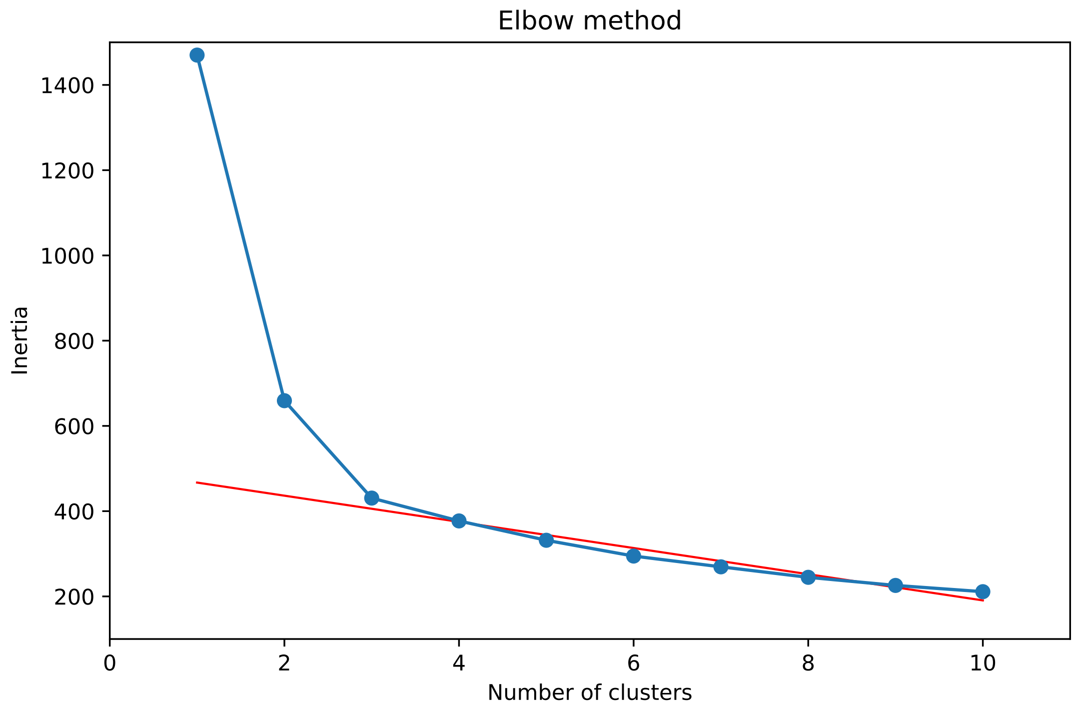

Figure 1: Cluster analysis

3 Problem 3: Visualizing the classes

3.1 Pair Projection

Plotting each pair of features we can see how the different classes of seed are well differentiable in most of the plots, as it's visible in figure 2. The most interesting plot seems to be the one between C4 and C7, since it is the one where the classes overlap the less.

3.2 Gaussian Random Projection

Another interesting result is obtained with the Gaussian Random Projection (figure 3). In this case we distinguished 3 classes (red, green, blue) and two other samples (grey) that seem to be outliers, even though they belong to class 2.

3.3 UMAP Projection

In the UMAP projection (figure 4) the 2 outliers are even more evident (they would belong to the blue class). Probably the best thing to do would be to delete these samples from the dataset.

4 Problem 4: Evaluating clustering

Since there is no obvious linear border between the clusters a "best effort" approach will be executed. There is always the possibility that some data have flaws, and that may add to the complexity of the problem. There seems to be three clusters, but then there are outliers that blur the picture. The K-Means clustering function does a fairly good job in allocating

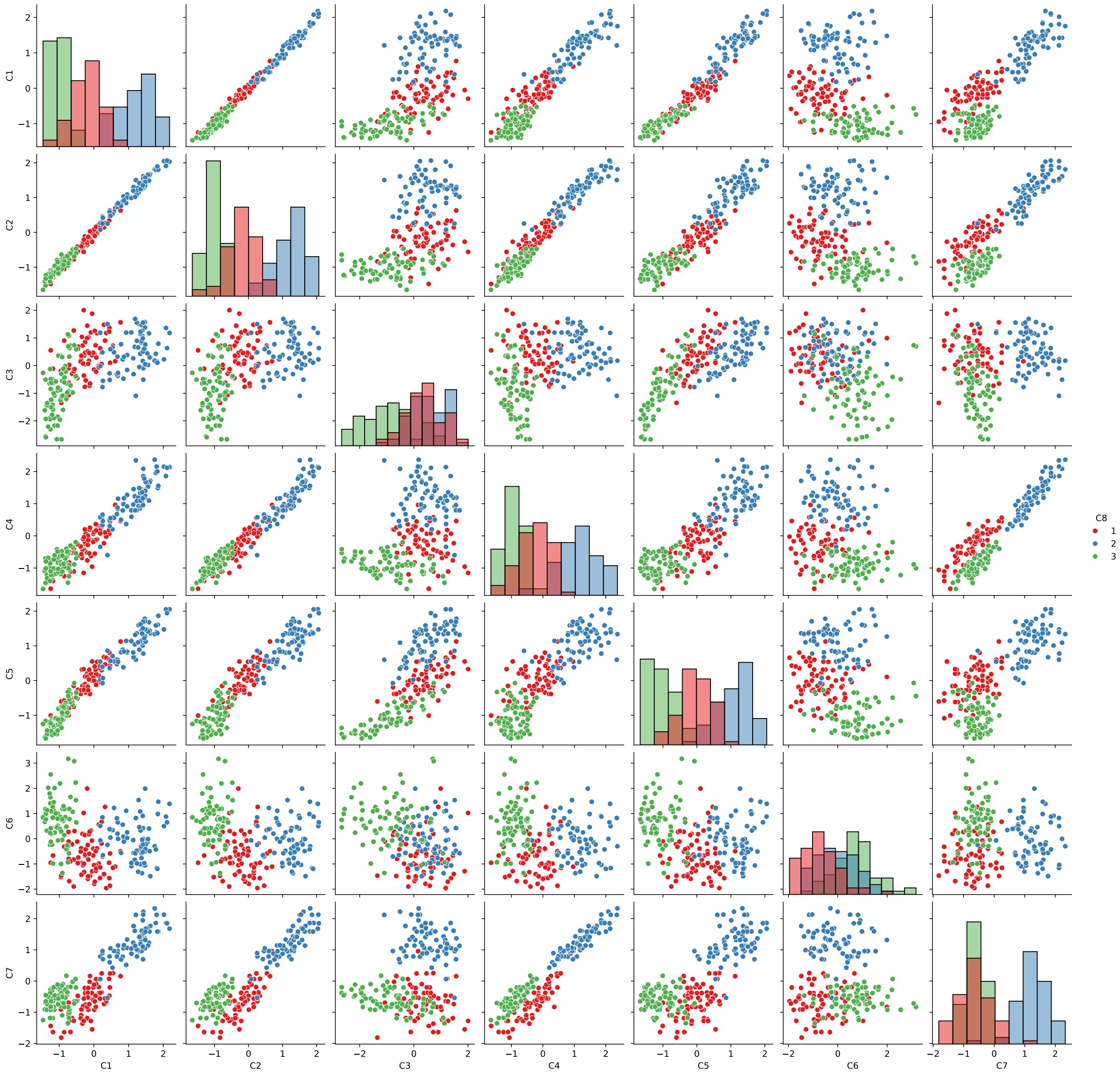

Figure 2: Pair Projection

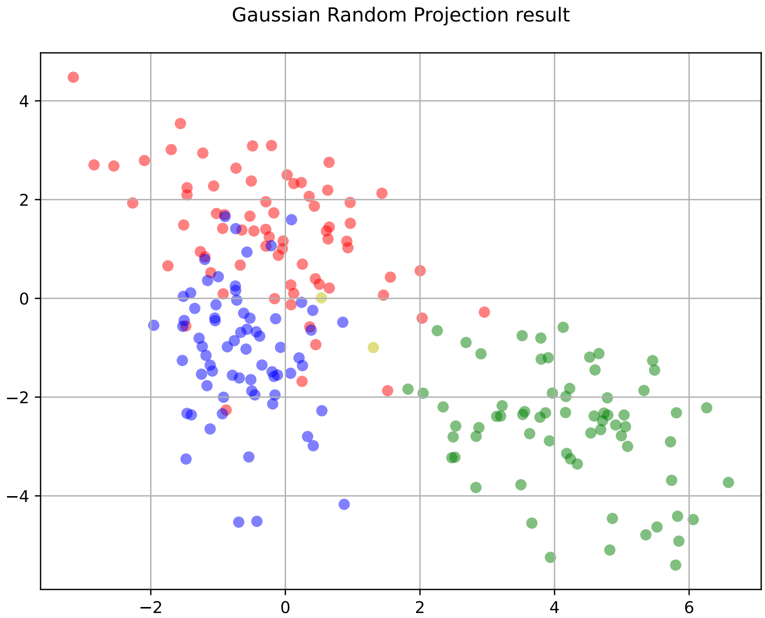

Figure 3: Gaussian Random Projection

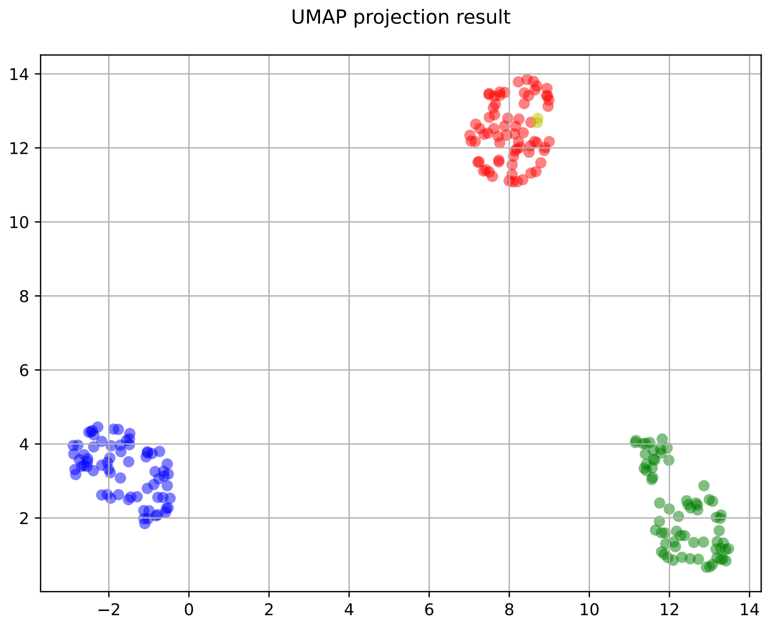

Figure 4: UMAP projection

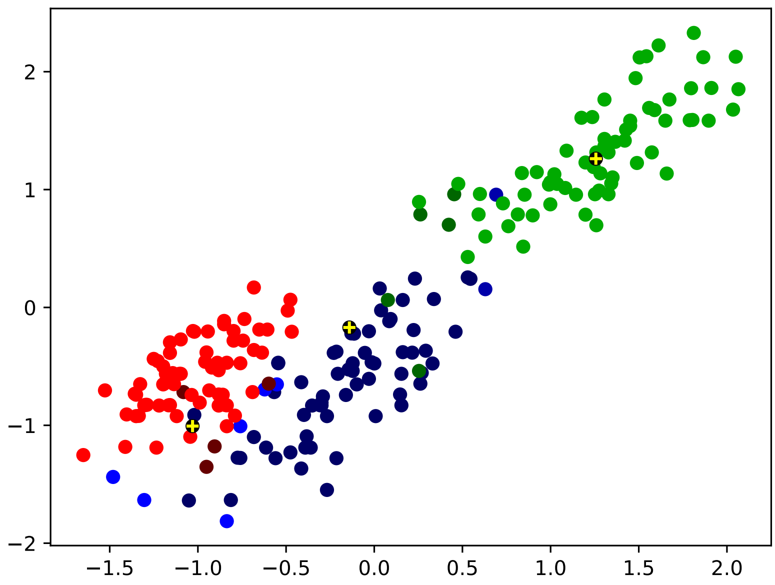

Figure 5: Clustering, with markings for cluster center

Accuracy score:	**0.9190**
Rand index:	**0.8997**

Table 2: Metrics for K-Mean clustering result

points to clusters. Given the class label data, and comparing it to the clusters there is an accuracy of **0.9190**, which counts as a quite good value. Trying to elevate that value might mean adapting to noise.

5 Problem 5: Agglomerative clustering

We used agglomerative clustering to compute a hierarchical clustering for the data and we then tried different linkage options.
The result is visible in figure 7. Looking at the plots it seems that the best work is done by the ward, the average and the complete linkage options.
To determine which one is the best one we looked at the accuracy of each linkage option. The worst is the single. The result was the following:

- Ward: 0.3926339709101015
- Complete: 0.35019845816108097

	Data labels		
	65	0	5
Predicted labels	0	66	4
	2	6	62

Table 3: Confusion matrix, se diagram 6

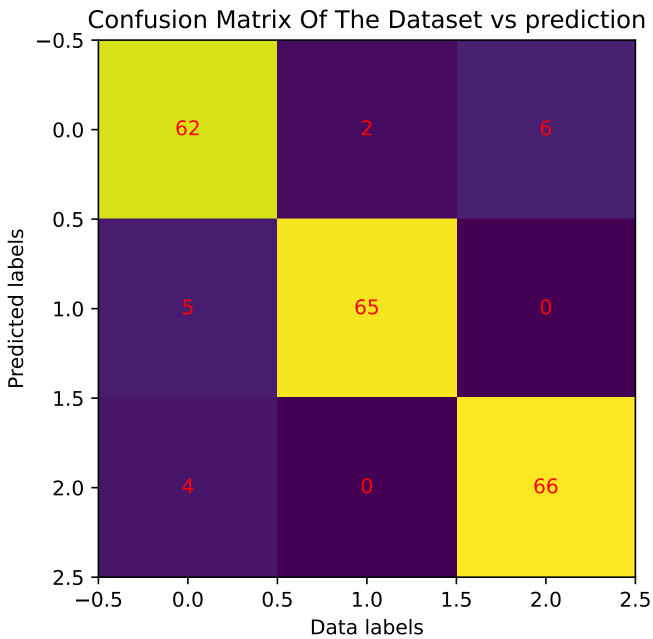

Figure 6: Confusion matrix showing number of accurate classification vs faulty classifications

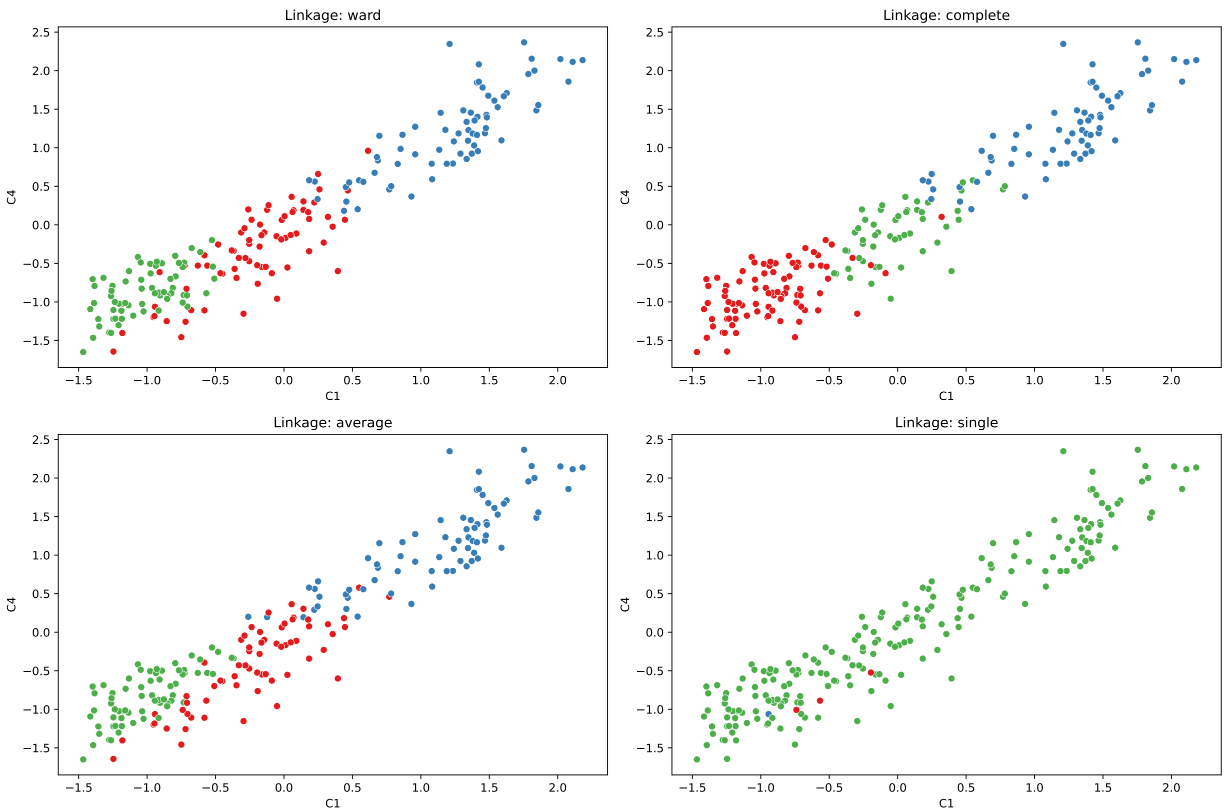

Figure 7: Agglomerative Clustering

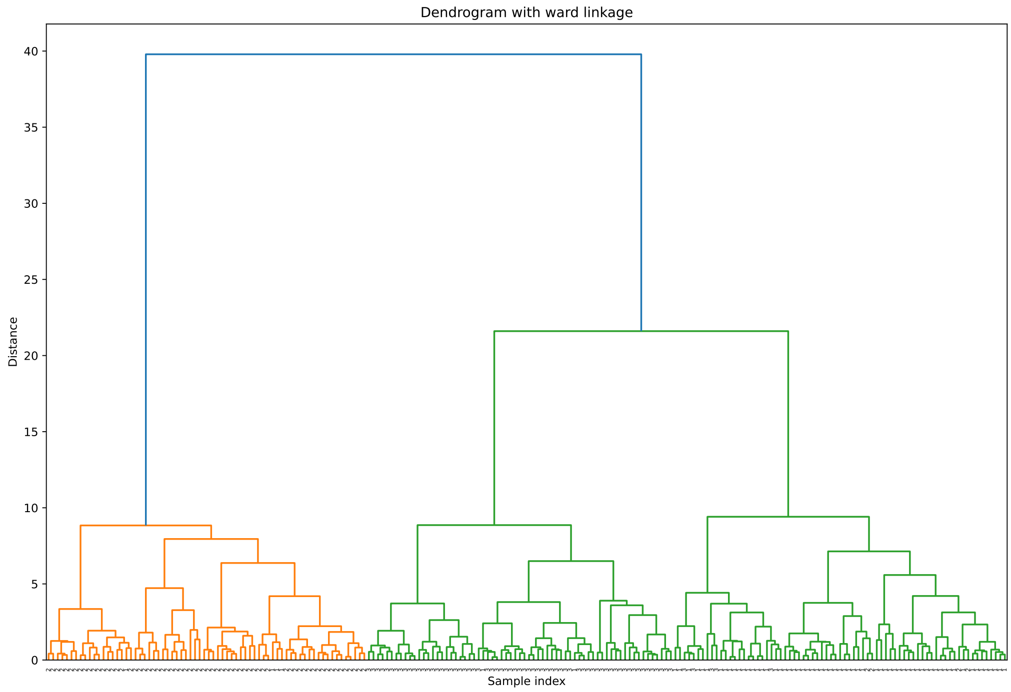

Figure 8: Dendrogram

- Average: 0.3759568059006467
- Single: -0.005642378923309307

Looking at the accuracy we determined that the absolute best one was 'ward' and we decided to choose that.
For that we plotted the dendrogram that is possible to see in figure 8.

A Source code

This is the complete listing of the source code.

A.1 Problem 1 and 2

Code for normalizing data and examine optimal number of clusters.

```
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-21
# Assignment 5
# Problem 1, 2
# Clustering
########################################################################
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.stats import zscore
from sklearn.linear_model import LinearRegression
########################################################################
filelocation_in = r"Uppgift05\\00_Uppg_Data\\" # folder to get data from
filelocation_out = r"Uppgift05\\test_data\\" # folder to write data to
## ####################################################################
df = pd.read_csv(filelocation_in + "seeds.tsv", sep = '\t', header=None)

## Problem 1, normalize data, Z-score chosen
df_zscore = df.apply(zscore) # Calculate Z-scores for each column
df_zscore.drop(7, axis='columns', inplace=True) # Drop class label

## Problem 2, find number of clusters
## Use elbow method to find nbr of clusters
inertias = []

for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(df_zscore)
    inertias.append(kmeans.inertia_)

ellbow = 3

x=[[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]]
regr = LinearRegression()
regr.fit(x[ellbow-1:11:], inertias[ellbow-1:11:])
k=regr.coef_
m=regr.intercept_
print("2 Residual sum of squares: %.2f"
      % np.mean((k*x[ellbow-1:11:]+m - inertias[ellbow-1:11:]) ** 2))

plt.figure(figsize=(8, 5))
plt.xlim(0,11)
plt.ylim(100,1500)
plt.plot(x[0:11:], k*x[0:11:]+m, color='red', linewidth=1)
plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.savefig(filelocation_out + 'plot_of_Elbow_method_problem2.pdf')
```

A.2 Problem 3

Code for Visualizing the classes.

```python
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-21
# Assignment 5
# Problem 3
# Visualizing the classes
#########################################################################
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import zscore
from sklearn.random_projection import GaussianRandomProjection
import seaborn as sns
import umap
#########################################################################
filelocation_in = r"Uppgift05\\00_Uppg_Data\\" # folder to get data from
filelocation_out = r"Uppgift05\\test_data\\" # folder to write data to
## ######################################################################
df = pd.read_csv(filelocation_in + "seeds.tsv", sep = '\t', header=None)
df.columns = ['C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8']
colormap = np.array(['r', 'g', 'b', 'y'])

## Problem 1, normalize data, Z-score chosen
df_zscore = df.apply(zscore) # Calculate Z-scores for each column
df_zscore['C8'] = df['C8']
df_zscore['C8'] = np.where(df_zscore['C8']==4, 2, df_zscore['C8'])

# Select only the numerical columns for plotting
numerical_columns = ['C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7']
data_for_plot = df_zscore[numerical_columns + ['C8']]

# Plot scatter plots between each pair of features,
# coloring the points by the class label
sns.pairplot(data_for_plot, hue='C8', palette='Set1', diag_kind='hist')
plt.savefig(filelocation_out + 'Pair Projection result.pdf')

# df_zscore.drop(['C8'], axis='columns', inplace=True) # Drop class label
# Extract numerical features from the DataFrame
X = df_zscore.values

# Apply Gaussian Random Projection to project the data to two dimensions
random_projection = GaussianRandomProjection(n_components=2,
                                             random_state=42)
X_projected = random_projection.fit_transform(X)

# Plot the projected data as a scatter plot
plt.figure(figsize=(8, 6))
scatter=plt.scatter(
    X_projected[:, 0], X_projected[:, 1],
    c=colormap[df['C8']-1], alpha=0.5)
plt.title('Gaussian Random Projection result\n')
plt.grid(True)
plt.savefig(filelocation_out + 'Gaussian Random Projection result.pdf')

## Apply UMAP
reducer = umap.UMAP(n_components=2)
```

```python
embedding = reducer.fit_transform(X)
plt.figure(figsize=(8, 6))
plt.scatter(
    embedding[:, 0], embedding[:, 1],             # type: ignore
    c=colormap[df['C8']-1], alpha=0.5)
plt.title('UMAP projection result\n')
plt.grid(True)
plt.savefig(filelocation_out + 'UMAP Projection result.pdf')
```

A.3 Problem 4

Code for Evaluating clustering.

```python
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-21
# Assignment 5
# Problem 4
# Evaluating Clustering
########################################################################
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.stats import zscore
from sklearn.metrics import rand_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
########################################################################
filelocation_in = r"Uppgift05\\00_Uppg_Data\\" # folder to get data from
filelocation_out = r"Uppgift05\\test_data\\" # folder to write data to
## ####################################################################
df = pd.read_csv(filelocation_in + "seeds.tsv", sep = '\t', header=None)
colormap = np.array(['#000066', '#0000aa', '#0000ff', # Blue
                     '#006600', '#00aa00', '#00ff00', # Green
                     '#660000', '#aa0000', '#ff0000', # Red
                     '#FFFF00'])                      # Yellow

## Normalize data with Z-scores and extract labels
df_zscore = df.apply(zscore) # Calculate Z-scores for each column
df_zscore.drop(7, axis='columns', inplace=True) # Drop class label
labels = df[7] # classification labels in data

## Problem 4, find 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0).fit(df_zscore)

# Set appropriate permutation of labels
km_lbl = kmeans.labels_
for i in range(len(km_lbl)):
    km_lbl[i] = (km_lbl[i] + 1) % 3 + 1

## Calculate metrics
cm = confusion_matrix(labels, km_lbl)
print(cm)

## Compute accuracy
print('Accuracy score: ', accuracy_score(labels, km_lbl))
## Compute Rand index
print('Rand index: ', rand_score(labels, km_lbl))
```

```

plt.clf
plt.scatter(df_zscore[1], df_zscore[6],
            c=colormap[3 * (df[7] - 1) + (km_lbl - 1)])
plt.scatter(kmeans.cluster_centers_[:, 0],
            kmeans.cluster_centers_[:, 1], c='#000000', marker='o')
plt.scatter(kmeans.cluster_centers_[:, 0],
            kmeans.cluster_centers_[:, 1], c='#FFFF00', marker='+')
plt.savefig(filelocation_out + 'P4_clusters.pdf')
plt.show()
plt.clf
for i in range(3):
    for j in range(3):
        plt.annotate(str(round(cm[i][j], 2)),
                     xy=(j, i),
                     ha='center', va='center', color='red')
plt.title("Confusion Matrix Of The Dataset vs prediction")
plt.xlabel("Data labels")
plt.ylabel("Predicted labels")
plt.imshow(cm,  interpolation='nearest')
plt.savefig(filelocation_out + 'P4_confusion_matrix.pdf')
plt.show()
```

A.4 Problem 5

Code for Agglomerative clustering.

```
## PROBLEM 5

import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
import seaborn as sns
from sklearn.metrics import silhouette_score


## Agglomerative clustering

# Extract predictors (features) and dependent variable from df_zscore
X = df_zscore[['C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7']]
y = df_zscore['C8']

# Initialize a dictionary to store linkage options and corresponding clusterings
linkage_options = {'ward': AgglomerativeClustering(n_clusters=3, linkage='ward'),
                   'complete': AgglomerativeClustering(n_clusters=3, linkage='complet
                   'average': AgglomerativeClustering(n_clusters=3, linkage='average
                   'single': AgglomerativeClustering(n_clusters=3, linkage='single')}

# Perform hierarchical clustering with different linkage options
plt.figure(figsize=(15, 10))
for i, (linkage, clustering) in enumerate(linkage_options.items(), 1):
    plt.subplot(2, 2, i)
    clustering.fit(X)
    sns.scatterplot(x='C1', y='C4', hue=clustering.labels_, data=df_zscore, palette=
    plt.title(f'Linkage: {linkage}')
    plt.xlabel('C1')
    plt.ylabel('C4')
plt.tight_layout()
plt.savefig('AgglomerativeClustering.pdf')
```

```python
plt.show()


## Calculate the accuracy for each linkage option

# Initialize a dictionary to store silhouette scores for each linkage option
silhouette_scores = {}

# Compute silhouette score for each linkage option
for linkage, clustering in linkage_options.items():
    clustering.fit(X)
    silhouette_scores[linkage] = silhouette_score(X, clustering.labels_)

# Print silhouette scores for each linkage option
for linkage, score in silhouette_scores.items():
    print(f"Silhouette Score for {linkage}: {score}")


## Plot the dendrogram

clustering = AgglomerativeClustering(n_clusters=3, linkage='ward')
clustering.fit(X)

Z = linkage(X, method=best_linkage)

plt.figure(figsize=(15, 10))
dendrogram(Z, labels=y.values)
plt.title(f'Dendrogram with {best_linkage} linkage')
plt.ylabel('Distance')
plt.savefig('Dendrogram.pdf')
plt.show()
```