Part 2: Coding Task

EE214 - Assignment 2
(2025 Spring)

May 5, 2025

1 Overview

Part 2 of this assignment is a self-contained exploration of classical clustering techniques on real-world data. You will implement several algorithms, evaluate them with internal validation metrics, and reflect on metric agreement, cluster structure, and practical relevance. You are expected to use AI tools (e.g., ChatGPT, GitHub Copilot) only as an aid. You must document all AI assistance in your final report.

2 Clustering Study on the Dermatology Dataset & an Unlabelled Feature Set

2.1 Objective

Evaluate and compare multiple clustering techniques—k-Means, Agglomerative, DBSCAN on the Dermatology dataset. Using an AutoEncoder, extract the latent features, and repeat the clustering to observe how the resulting clusters differ from those on the raw features. Determine an appropriate number of clusters (or DBSCAN parameters) using appropriate internal scores (e.g. Elbow method and Silhouette scores), analyse how these metrics agree or conflict, and interpret what the results reveal about cluster structure and practical usefulness. Finally, apply the same workflow to the **unlabelled feature set** supplied by the instructors and report the most plausible cluster configuration for that data.

2.2 Data Sources

The datasets used for this assignments originate from UCI Repository as follows:

- **Dermatology dataset** — UCI Machine-Learning Repository `https://archive.ics.uci.edu/ml/datasets/Dermatology`
- **Unknown dataset** — a 5450×10 feature matrix extracted from another public UCI dataset. The raw samples were pre-processed with PCA (10 principal components), and the resulting CSV file is provided. Ground-truth class labels are withheld.

2.3 Steps and Requirements

1. **Data Loading and Pre-processing**

   - Load the Dermatology dataset and given dataset(unknown_dataset.csv) with `pandas`.
   - Handle missing values if present.
   - Apply standard scaling; you may compare with Min-Max scaling and discuss any differences. (Note: Skip this step and 2.4.2 for the unknown dataset)

2. **Feature Extraction**

   (a) PCA — retain enough components to explain $\geq 95\%$ of the variance (report the exact number).

   (b) AutoEncoder — design a shallow encoder–decoder that compresses the data to a (n-dimensional latent space; train until loss plateaus. (explain how you decided the value of n). The choice of optimizer, loss function, and hyperparameters is left to your discretion.

3. **Clustering Algorithms**

   (a) k-Means — run for $k = 2 \ldots 15$ with `random_state=42`.

   (b) Agglomerative — same k range; generate a linkage dendrogram (use at least three linkage methods.)

   (c) DBSCAN — grid-search at least 10 *eps* values and *minPts*.

   Apply each algorithm in the case of Dermatology dataset on the *raw* features, *PCA* feature space, and the *AutoEncoder* latent space.

4. **Validation Metrics & Visualisation**

   - Compute *Elbow inertia* and *Silhouette score* for every k (raw, PCA, AE).
   - Plot Elbow curves and Silhouette score line/bar charts.
   - For Agglomerative, provided the dendrogram plot and corresponding Silhouette scores.
   - For DBSCAN, create a Silhouette heat-map over the (*eps*, *minPts*) grid.
   - Produce 2-D scatter plots (PCA or t-SNE) coloured by the final cluster labels for both datasets.

5. **Cluster-Count / Parameter Decision**

   - Using the metrics above, justify a "best" k (or DBSCAN parameter pair) for the Dermatology dataset.
   - Repeat the decision process for the unknown dataset.

6. **Analysis Tasks (include in report)**

   - **Correct Cluster-Count Selection & Clear Plot (5 pts)** — Present your final configuration—k for k-Means/Agglomerative or *eps*/*minPts* for DBSCAN—and include a well-labelled 2-D scatter plot (or comparable visual) that clearly shows the clusters.

- **Metric Agreement/Conflict Discussion (5 pts)** — explain where Elbow and Silhouette (or dendrogram, DBSCAN grid) align or diverge, and why.
- **Representation Effect (5 pts)** Compare clustering quality across the *raw*, *PCA*, and *AutoEncoder* feature spaces using Silhouette scores and visual inspection of cluster plots. Which representation produced the clearest separation?
- **Stability Assessment (5 pts)** Run a selected algorithm e.g. k-Means several times with different random seeds and perform clustering. Comment on how stable the clustering is and what that implies about its reliability.
- **Alternative Approaches or Improvements (5 pts)** — Identify a key weakness that you think prevents the current pipeline from producing stable or high-quality clusters, then implement one concrete change to address it. Possible options include, but are not limited to, implement an advanced clustering algorithm, adding new validation metrics such as Davies–Bouldin, Calinski–Harabasz, or Adjusted Rand Score. Provide a brief rationale for your chosen modification and report the results.

Deliverables

- A single Jupyter notebook with code, metrics, and plots for both datasets.
- A concise report answering the analysis tasks and fully documenting AI usage.

2.4 Code Skeleton Example

Below is a high-level Python skeleton that mirrors Part 2 requirements. Feel free to modify the structure, but keep the same logical order so your work is easy for the graders to follow.

```
# ========================================
# EE214    Assignment 2     Clustering Skeleton
# ========================================
# Code skeleton for Task 2. You may change it as you deem necessary.

# ---------- 0. Imports ----------
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from ucimlrepo import fetch_ucirepo          #  o n e line  UCI fetch
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.metrics import silhouette_score

# For AutoEncoder
import torch, torch.nn as nn, torch.optim as optim

# ---------- 1. Data Loading ----------
derm = fetch_ucirepo(name="Dermatology")          # TODO: cache locally if slow
X_raw = derm.data.features.to_numpy()            # 34 features (no labels)
X_unknown = pd.read_csv("unknown_dataset.csv").to_numpy()  # 5452 10  PCA
```

```
25
26 # ---------- 2. Scaling ----------
27 std_scaler    = StandardScaler()
28 X_std         = std_scaler.fit_transform(X_raw)
29
30 minmax_scaler = MinMaxScaler()                    # for comparison
31 X_minmax      = minmax_scaler.fit_transform(X_raw)
32
33 # ---------- 3. Dimensionality Reduction ----------
34 pca = PCA(n_components=0.95, random_state=42)
35 X_pca = pca.fit_transform(X_std)
36
37 # --- AutoEncoder stub ------------------------------------------------
38 class AE(nn.Module):
39     def __init__(self, in_dim, latent_dim=10):
40         super().__init__()
41         # TODO define encoder / decoder layers
42     def forward(self, x):
43         # TODO forward pass
44         return x
45
46 def train_ae(x_np, latent_dim, epochs):
47     """Returns latent representation (np.ndarray)."""
48     # TODO: build model, optimizer, loss, training loop
49     return x_np                                    # placeholder
50
51 latent_dim =
52 epochs =
53 X_ae = train_ae(X_std, latent_dim, epochs)
54 # ----------------------------------------------------------------------
55
56 # ---------- 4. Helper:  k Means  Elbow & Silhouette ----------
57 def elbow_silhouette_plot(X, k_range=range(2, 6)):
58     inertia, sil = [], []
59     for k in k_range:
60         # TODO : run KMeans, get inertia and Silhouette scores
61     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,4))
62     ax1.plot(k_range, inertia, 'bo-');  ax1.set_title("Elbow")
63     ax2.plot(k_range, sil, 'ro-');      ax2.set_title("Silhouette")
64     plt.show()
65     # TODO: select best k by inspecting the plots
66
67 # ---------- 5. Helper: Agglomerative dendrogram ----------
68 from scipy.cluster.hierarchy import linkage, dendrogram
69 def dendro_plot(X, method="ward"):    # TODO : use other linkage methods for
       comparison
70
71     Z =     # TODO : get linkage
72     plt.figure(figsize=(8,3))
73     # TODO : plot dendogram
74     plt.title(f"Dendrogram ({method})")
75     plt.show()
76     # TODO: compute Silhouette for k=2..15
77
78 # ---------- 6. Helper: DBSCAN grid  h e a t map  ----------
79 def dbscan_heatmap(X, eps_vals, min_samples_vals):
```

```
    """
    High level outline:
      1. Iterate over every (eps, min_samples) combination.
      2. Fit DBSCAN, record:
             number of clusters (ignore noise label -1)
             Silhouette score (set to NaN if clusters < 2)
      3. Pivot the results into two matrices:
             heatmap of cluster counts
             heatmap of Silhouette scores
      4. Display the two heatmaps side-by-side.
      5. After inspecting the visuals, manually choose the
         best (eps, min_samples) pair for your final model.
    """
    # TODO: implement the loop, DataFrame creation, seaborn heatmaps,
    #       and return or log the chosen parameters.
    pass


# ---------- 7. Final clustering (student-chosen params) ----------
# TODO: run KMeans, Agglomerative, DBSCAN with the parameters you selected
# labels_km  =
# labels_agg =
# labels_db  =

# ---------- 8. Visualisation helper ----------
def scatter_2d(X2, labels, title):
    plt.figure(figsize=(4,4))
    plt.scatter(X2[:,0], X2[:,1], c=labels, cmap="tab10", s=8)
    plt.title(title); plt.show()

# Example: scatter_2d(X_pca, labels_km, "Dermatology: KMeans on PCA")

# ---------- 9. UNKNOWN DATASET ----------
# Repeat steps 4-8 for X_unknown (skip AE training).
```

Listing 1: Python Code Skeleton for Part2

3 Submission Guidelines for Part 2

- Submit a complete Jupyter Notebook containing your code, plots, and experimental results. Ensure the notebook runs end-to-end with a fixed random seed (where applicable) for reproducibility.
- Provide a written report that includes answers to the analysis questions, along with a thoughtful reflection on your work for Part 2.
- Clearly document every instance of AI assistance used (if any). Include which tool was used, what you asked, and how you integrated the response.

Good luck and happy coding!