Clustering Analysis Report on E. coli Dataset

# 1. Introduction

This report is about clustering the E. coli dataset using PCA and both K-Means and Agglomerative Clustering methods. The goal is to analyze how well clustering groups match the original labels and to visualize the clusters in 2D space.

# 2. Methodology

- Scaled the data using StandardScaler.

- Reduced dimensions using PCA (2 components).

- Used K-Means with 8, 10, and 15 clusters.

- Applied Agglomerative Clustering with various linkages and thresholds.

**Scripts:**

from sklearn.cluster import KMeans, AgglomerativeClustering

import numpy as np

import matplotlib.pyplot as plt

import sklearn

from sklearn import datasets

from sklearn import model\_selection

from sklearn.decomposition import PCA

from random import sample

import pandas as pd

# Load dataset

df = pd.read\_csv('D:/AI/labwork\_2/39\_Ecoli/ecoli.data', sep=r'\s+', header=None)

X = df.iloc[:, 1:-1] # Skips the first column (ID) and selects features

y = df.iloc[:, -1] # Last column as target

scaler = sklearn.preprocessing.StandardScaler()

X = scaler.fit\_transform(X)

pca = PCA(n\_components = 2)

pca.fit(X, 2)

X\_pca = pca.transform(X)

unniq\_labels = np.unique(y)

nlabels = len(unniq\_labels)

Ncolors = nlabels

# Choose a colormap and generate Ncolors distinct colors

cmap = plt.get\_cmap("rainbow")

colors = cmap(np.linspace(0, 1, Ncolors))

plt.figure(1)

for i,l in enumerate(unniq\_labels):

idxs = np.where(y==l)[0]

plt.scatter(X\_pca[idxs,0], X\_pca[idxs,1], c=colors[i])

plt.legend(unniq\_labels)

nclusters = 10

print('nclusters:', nclusters)

clustering1 = KMeans(nclusters)

clusters = clustering1.fit\_predict(X)

colors = cmap(np.linspace(0, 1, nclusters))

plt.figure(2)

for i,c in enumerate(clusters):

plt.scatter(X\_pca[i,0], X\_pca[i,1], c=colors[c])

plt.legend(np.arange(nclusters))

Ns = np.zeros((nclusters,), dtype=np.int32)

for i in range(nclusters):

Ns[i] = np.sum(clusters == i)

print("Elements in each cluster:", Ns)

nlabels = len(unniq\_labels)

label\_to\_index = {label: idx for idx, label in enumerate(unniq\_labels)}

y\_int = y.map(label\_to\_index)

NNs = np.zeros((nlabels, nclusters), dtype=np.int32)

for t, p in zip(y\_int, clusters):

NNs[t, p] += 1

print("Cluster stats:\n" , NNs)

plt.show()

# 3. Cluster Composition Tables

**\*\*K-Means with 10 Clusters\*\***

[40 68 34 1 9 19 35 30 34 66]

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect. A screen shot of a graph

AI-generated content may be incorrect.

**\*\*Agglomerative with 12 Clusters\*\***

[[ 0 34 1 ... 48 33] A screenshot of a computer code

AI-generated content may be incorrect. A screenshot of a computer program

AI-generated content may be incorrect. A screen shot of a graph

AI-generated content may be incorrect.

**\*\*Agglomerative with 15 Clusters\*\***

[[32 1 8 ... 0 1 1]

**Scripts:**

nclusters = 15

print('nclusters:', nclusters)

clustering1 = KMeans(nclusters)

clusters = clustering1.fit\_predict(X)

colors = cmap(np.linspace(0, 1, nclusters))

plt.figure(4)

for i,c in enumerate(clusters):

plt.scatter(X\_pca[i,0], X\_pca[i,1], c=colors[c])

plt.legend(np.arange(nclusters))

Ns = np.zeros((nclusters,), dtype=np.int32)

for i in range(nclusters):

Ns[i] = np.sum(clusters == i)

print("Elements in each cluster:", Ns)

nlabels = len(unniq\_labels)

label\_to\_index = {label: idx for idx, label in enumerate(unniq\_labels)}

y\_int = y.map(label\_to\_index)

NNs = np.zeros((nlabels, nclusters), dtype=np.int32)

for t, p in zip(y\_int, clusters):

NNs[t, p] += 1

print("Cluster stats:\n" , NNs)

plt.show()

A screenshot of a computer code

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a graph

AI-generated content may be incorrect.

**\*\*Agglomerative (linkage = 'average')\*\***

**Scripts:**

from sklearn.cluster import KMeans, AgglomerativeClustering

import numpy as np

import matplotlib.pyplot as plt

import sklearn

from sklearn import datasets

from sklearn import model\_selection

from sklearn.decomposition import PCA

from random import sample

import pandas as pd

# Load dataset

df = pd.read\_csv('D:/AI/labwork\_2/39\_Ecoli/ecoli.data', sep=r'\s+', header=None)

X = df.iloc[:, 1:-1] # Skips the first column (ID) and selects features

y = df.iloc[:, -1] # Last column as target

scaler = sklearn.preprocessing.StandardScaler()

X = scaler.fit\_transform(X)

pca = PCA(n\_components = 2)

pca.fit(X)

X\_pca = pca.transform(X)

unniq\_labels = np.unique(y)

nlabels = len(unniq\_labels)

Ncolors = nlabels

# Choose a colormap and generate Ncolors distinct colors

cmap = plt.get\_cmap("rainbow")

colors = cmap(np.linspace(0, 1, Ncolors))

plt.figure(1)

for i,l in enumerate(unniq\_labels):

idxs = np.where(y==l)[0]

plt.scatter(X\_pca[idxs,0], X\_pca[idxs,1], c=[colors[i]])

plt.legend(unniq\_labels)

print('AgglomerativeClustering, linkage: average')

clustering = AgglomerativeClustering(linkage='average',n\_clusters=None, distance\_threshold=45)

clustering.fit(X)

clusters = clustering.labels\_

nclusters = len(np.unique(clusters))

print('Number of clusters:', nclusters)

Ns = np.zeros((nclusters,), dtype=np.int32)

for i in range(nclusters):

Ns[i] = np.sum(clusters == i)

print("Elements in each cluster:", Ns)

NNs = np.zeros((nlabels, nclusters), dtype=np.int32)

# Convert string labels in y to integers

label\_to\_index = {label: idx for idx, label in enumerate(unniq\_labels)}

y\_indices = np.array([label\_to\_index[label] for label in y])

for t, p in zip(y\_indices, clusters):

NNs[t, p] += 1

print("Cluster stats:\n" , NNs)

colors = cmap(np.linspace(0, 1, nclusters))

plt.figure(5)

for i,c in enumerate(clusters):

plt.scatter(X\_pca[i,0], X\_pca[i,1], c=[colors[c]])

plt.legend(np.arange(nclusters))

plt.title('Linkage: average')

plt.show()

A screen shot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screen shot of a graph

AI-generated content may be incorrect.

**\*\*Agglomerative (linkage = 'single')\*\***

**Scripts:**

print('AgglomerativeClustering, linkage: single')

clustering = AgglomerativeClustering(linkage='single',n\_clusters=None, distance\_threshold=25)

clustering.fit(X)

clusters = clustering.labels\_

nclusters = len(np.unique(clusters))

print('Number of clusters:', nclusters)

Ns = np.zeros((nclusters,), dtype=np.int32)

for i in range(nclusters):

Ns[i] = np.sum(clusters == i)

print("Elements in each cluster:", Ns)

NNs = np.zeros((nlabels, nclusters), dtype=np.int32)

# Convert string labels in y to integers

label\_to\_index = {label: idx for idx, label in enumerate(unniq\_labels)}

y\_indices = np.array([label\_to\_index[label] for label in y])

for t, p in zip(y\_indices, clusters):

NNs[t, p] += 1

print("Cluster stats:\n" , NNs)

colors = cmap(np.linspace(0, 1, nclusters))

plt.figure(6)

for i,c in enumerate(clusters):

plt.scatter(X\_pca[i,0], X\_pca[i,1], c=colors[c])

plt.legend(np.arange(nclusters))

plt.title('Linkage: single')

plt.show()

A computer screen shot of a code

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screen shot of a graph

AI-generated content may be incorrect.

**\*\*Agglomerative (linkage = 'single')\*\***

**Scripts:**

print('AgglomerativeClustering, linkage: complete')

clustering = AgglomerativeClustering(linkage='complete',n\_clusters=None, distance\_threshold=65)

clustering.fit(X)

clusters = clustering.labels\_

nclusters = len(np.unique(clusters))

print('Number of clusters:', nclusters)

Ns = np.zeros((nclusters,), dtype=np.int32)

for i in range(nclusters):

Ns[i] = np.sum(clusters == i)

print("Elements in each cluster:", Ns)

NNs = np.zeros((nlabels, nclusters), dtype=np.int32)

# Convert string labels in y to integers

label\_to\_index = {label: idx for idx, label in enumerate(unniq\_labels)}

y\_indices = np.array([label\_to\_index[label] for label in y])

for t, p in zip(y\_indices, clusters):

NNs[t, p] += 1

print("Cluster stats:\n" , NNs)

colors = cmap(np.linspace(0, 1, nclusters))

plt.figure(6)

for i,c in enumerate(clusters):

plt.scatter(X\_pca[i,0], X\_pca[i,1], color=colors[c])

plt.legend(np.arange(nclusters))

plt.title('Linkage: complete')

plt.show()

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer screen

AI-generated content may be incorrect.

# 5. Cluster Visualizations

- Used PCA to plot clusters in 2D.

- Visualized true labels and clustering results.

- Each cluster has a unique color.

- Final visualization (9 clusters, complete linkage) shows better structure.

# 6. Results & Interpretation

- PCA helped visualize the structure of the data.

- K-Means with 8 clusters matched class labels best.

- 10 and 15 clusters added detail but fragmented the data more.

- Agglomerative clustering with 12 and 15 clusters produced meaningful splits.

- Linkage methods like 'average', 'single' created only 1 cluster.

- 'Complete' linkage with tuned threshold successfully produced 9 clusters.

# 7. Conclusion

- PCA + clustering provided good insights.

- K-Means is quick but sensitive to the number of clusters.

- Agglomerative is powerful if properly tuned.

- 'Complete' linkage with 9 clusters gave best final result.

- Clustering helps discover hidden structure even without labels.