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

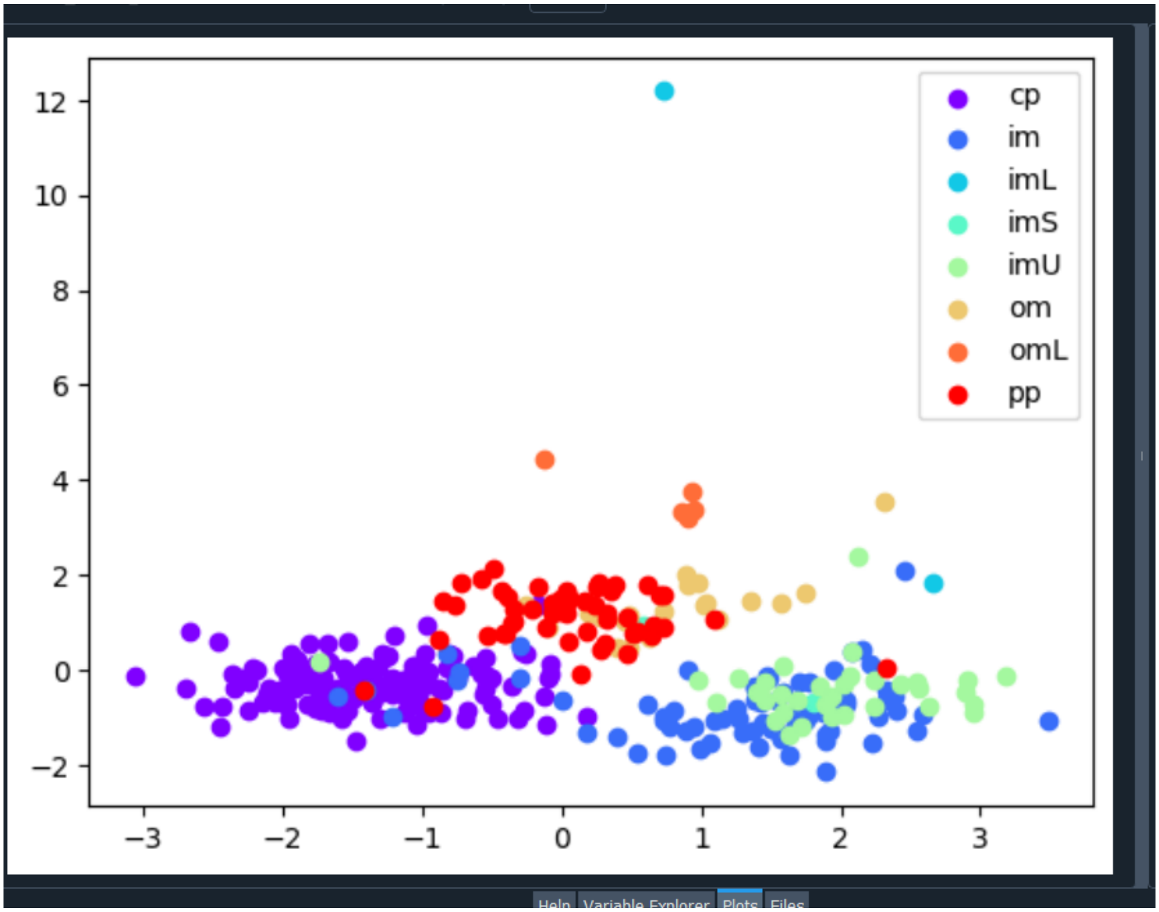
K-Means with 10 Clusters

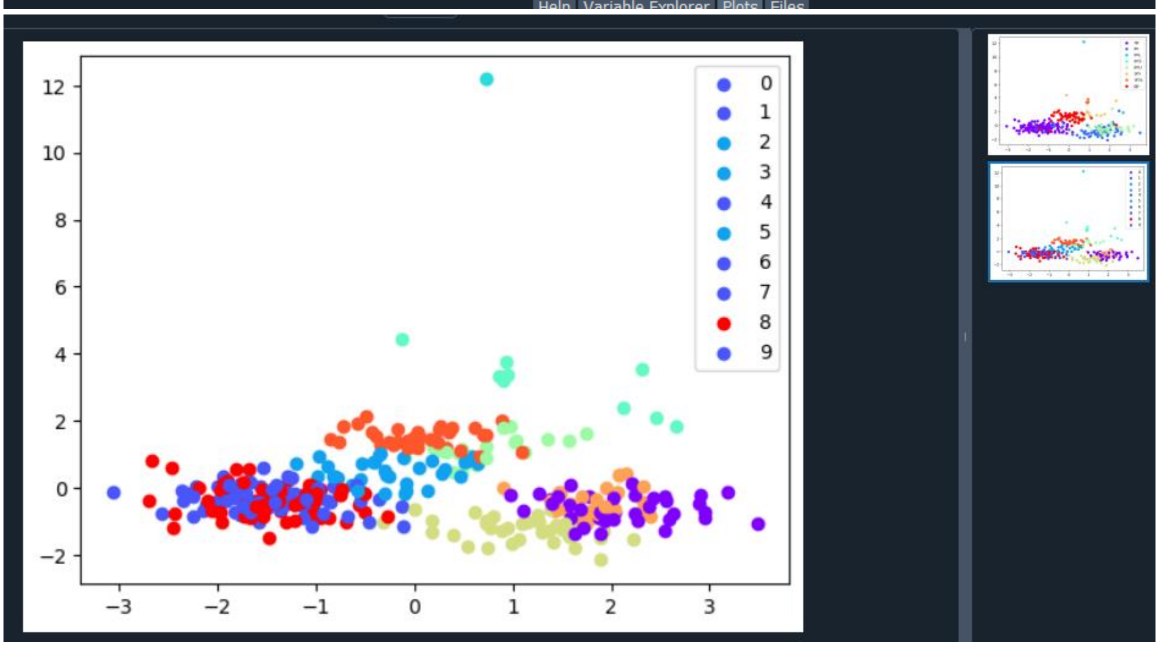
[40 68 34 1 9 19 35 30 34 66]

```
  warnings.warn(
d:\ai\labwork_2\assignment2.py:48: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_pca[i,0], X_pca[i,1], c=colors[c])
Elements in each cluster: [40 68 34  1  9 19 35 30 34 66]
Cluster stats:
 [[ 0 66 14  0  0  0  2  0  1 60]
 [13  0  3  0  1  0 32 23  0  5]
 [ 0  0  0  1  1  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  1  0  0]
 [27  0  0  0  1  0  1  5  0  1]
 [ 0  0  0  0  1 17  0  0  2  0]
 [ 0  0  0  0  5  0  0  0  0  0]
 [ 0  2 16  0  0  2  0  1 31  0]]
```

Name	Type	Size	Value
c	int32	1	8
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int32	(336,)	[1 1 2 ... 2 2 8]
colors	Array of float64	(10, 4)	[[5.00000000e-01 0.00000000e+00 1.00000000e+00 1.00000000e+00] [2.803 ...
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	9
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp': 7}
nclusters	int	1	10
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 10)	[[0 66 14 ... 0 1 60] [13 0 3 ... 23 0 5]
Ns	Array of int32	(10,)	[40 68 34 1 9 19 35 30 34 66]
p	int32	1	8
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module
scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module

Help Variable Explorer Plots Files

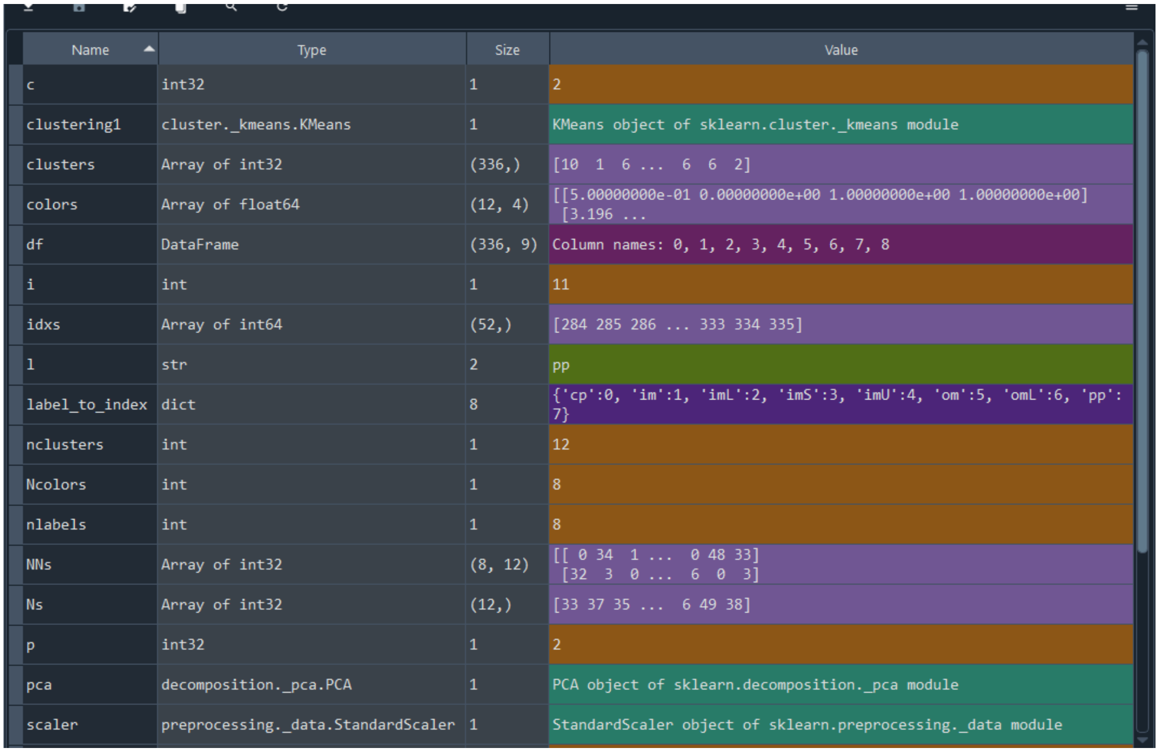




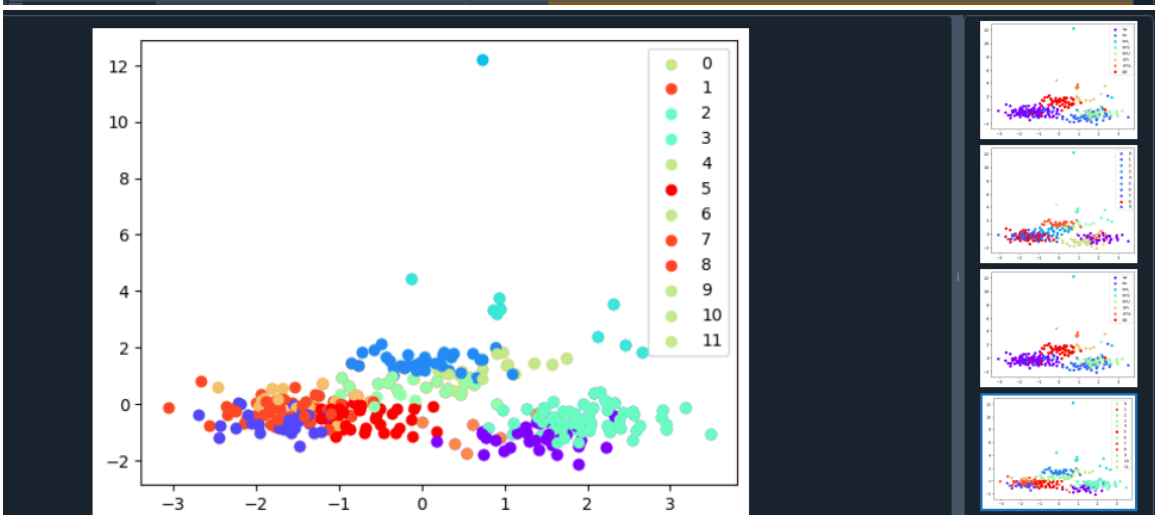
Agglomerative with 12 Clusters

[[0 34 1 ... 48 33]

```
RGBA value for all points.
  plt.scatter(X_pca[i,0], X_pca[i,1], c=colors[c])
nclusters: 12
Elements in each cluster: [33 37 35  1  9 63 26 19 20  6 49 38]
Cluster stats:
 [[ 0 34  1  0  0  0  9  0 18  0 48 33]
 [32  3  0  0  1 29  2  0  1  6  0  3]
 [ 0  0  0  1  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  1  0  0  0  0  0]
 [ 1  0  0  0  1 32  0  0  1  0  0  0]
 [ 0  0  2  0  1  0  0 17  0  0  0  0]
 [ 0  0  0  0  5  0  0  0  0  0  0  0]
 [ 0  0 32  0  0  1 14  2  0  0  1  2]]
```



Name	Type	Size	Value
c	int32	1	2
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int32	(336,)	[10 1 6 ... 6 6 2]
colors	Array of float64	(12, 4)	[[5.00000000e-01 0.00000000e+00 1.00000000e+00 1.00000000e+00] [3.196 ...
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	11
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp': 7}
nclusters	int	1	12
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 12)	[[0 34 1 ... 0 48 33] [32 3 0 ... 6 0 3]]
Ns	Array of int32	(12,)	[33 37 35 ... 6 49 38]
p	int32	1	2
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module
scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module



Agglomerative with 15 Clusters

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

Scripts:

nclusters = 15

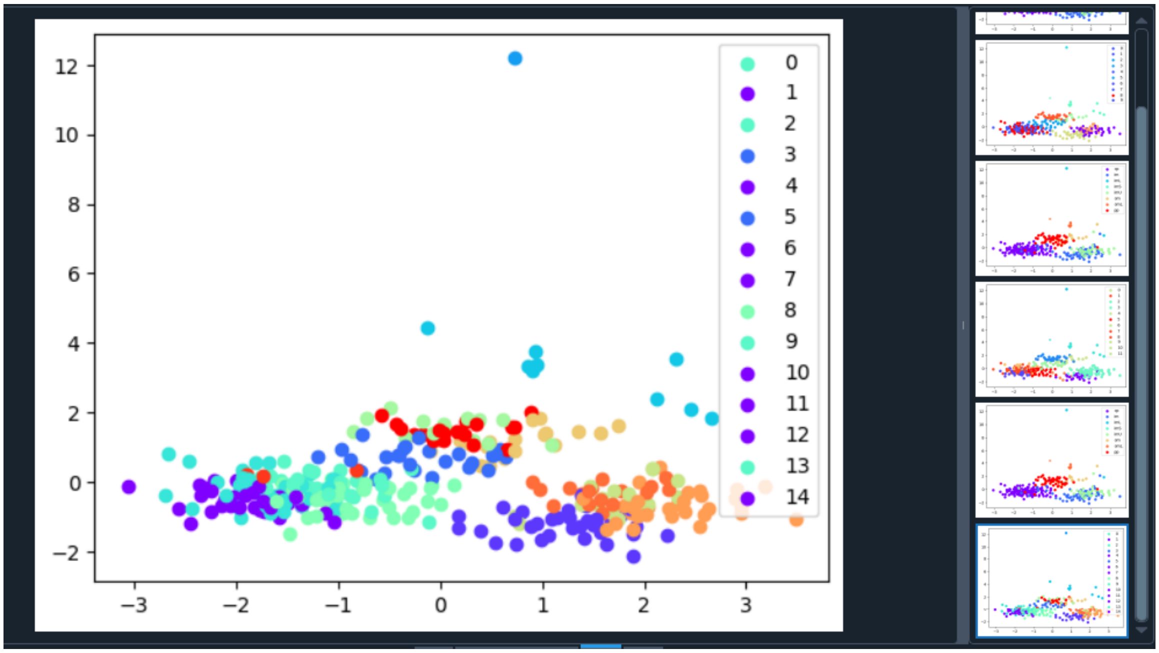
print('nclusters:', nclusters)

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()
```

```
use the 'color' keyword-argument or provide a 2D array with a single row if you intend to spec
RGBA value for all points.
  plt.scatter(X_pca[i,0], X_pca[i,1], c=colors[c])
nclusters: 15
Elements in each cluster: [33 34 27  1  9 40 33 36 14 19 18 33 17  3 19]
Cluster stats:
 [[32  1  8  0  0 40 33 27  0  0  0  0  0  1  1]
 [ 0 32  1  0  1  0  0  7  0 16  0 14  5  1  0]
 [ 0  0  0  1  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  1  0  0  1  0  0  0  0  2  0 18 12  1  0]
 [ 0  0  0  0  1  0  0  0  0  0 17  0  0  0  2]
 [ 0  0  0  0  5  0  0  0  0  0  0  0  0  0  0]
 [ 1  0 17  0  0  0  0  2 14  1  1  0  0  0 16]]

In [5]:
```

Name	Type	Size	Value
c	int32	1	8
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int32	(336,)	[6 0 6 ... 2 2 8]
colors	Array of float64	(15, 4)	[[5.00000000e-01 0.00000000e+00 1.00000000e+00 1.00000000e+00] [3.588 ...
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	14
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp': 7}
nclusters	int	1	15
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 15)	[[32 1 8 ... 0 1 1] [0 32 1 ... 5 1 0]
Ns	Array of int32	(15,)	[33 34 27 ... 17 3 19]
p	int32	1	8
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module
scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module



Agglomerative (linkage = 'average')

<u>Scripts:</u>

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

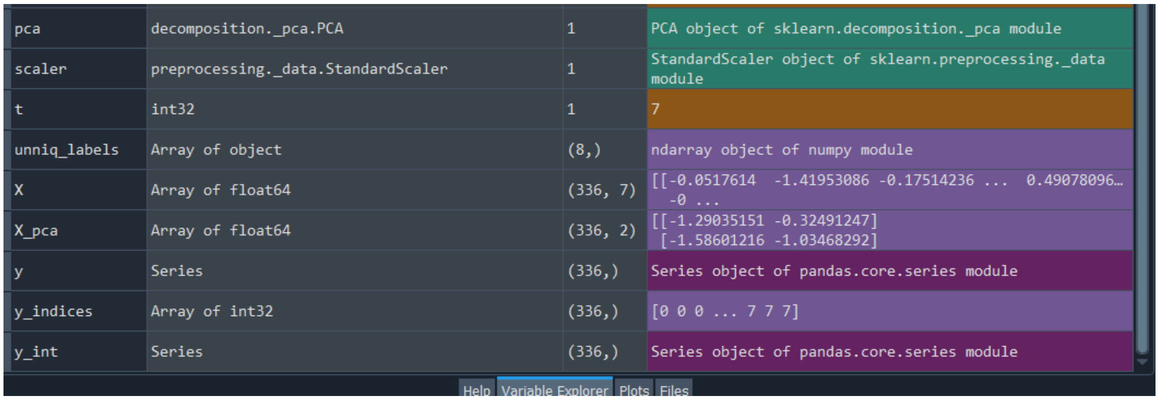
df = pd.read_csv('D:/AI/labwork_2/39_Ecoli/ecoli.data', sep=r'\s+', header=None)

```
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')
```

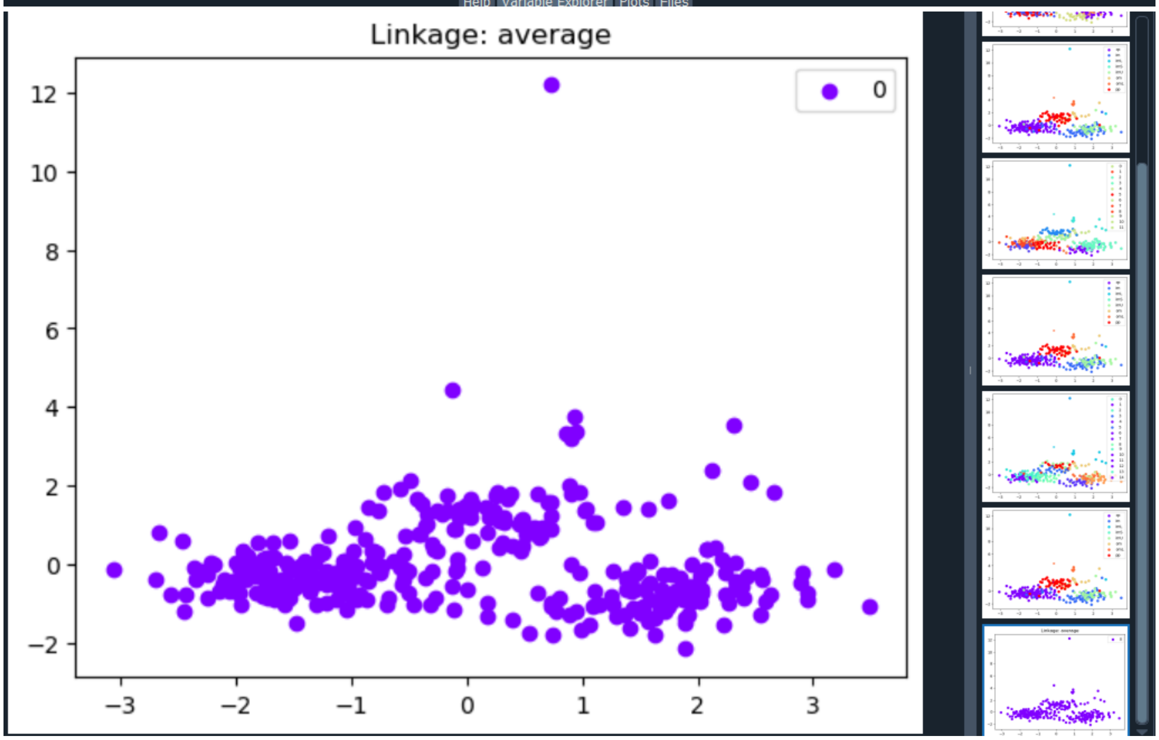
```
clustering = AgglomerativeClustering(linkage='average',n_clusters=None,
distance_threshold=45)

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()
```

```
In [7]: runfile('D:/AI/labwork_2/assignment2.py', wdir='D:/AI/labwork_2')
AgglomerativeClustering, linkage: average
Number of clusters: 1
Elements in each cluster: [336]
Cluster stats:
 [[143]
 [ 77]
 [  2]
 [  2]
 [ 35]
 [ 20]
 [  5]
 [ 52]]

In [8]:
```

Name	Type	Size	Value
c	int64	1	0
clustering	cluster._agglomerative.AgglomerativeClustering	1	AgglomerativeClustering object of sklearn.cluster._agglomerative modul ...
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int64	(336,)	[0 0 0 ... 0 0 0]
colors	Array of float64	(1, 4)	[[0.5 0. 1. 1.]]
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	335
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp':7}
nclusters	int	1	1
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 1)	[[143] [77]
Ns	Array of int32	(1,)	[336]
p	int64	1	0
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module



pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module
scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module
t	int32	1	7
unniq_labels	Array of object	(8,)	ndarray object of numpy module
X	Array of float64	(336, 7)	[[-0.0517614 -1.41953086 -0.17514236 ... 0.49078096… -0 ...
X_pca	Array of float64	(336, 2)	[[-1.29035151 -0.32491247] [-1.58601216 -1.03468292]
y	Series	(336,)	Series object of pandas.core.series module
y_indices	Array of int32	(336,)	[0 0 0 ... 7 7 7]
y_int	Series	(336,)	Series object of pandas.core.series module



Agglomerative (linkage = 'single')

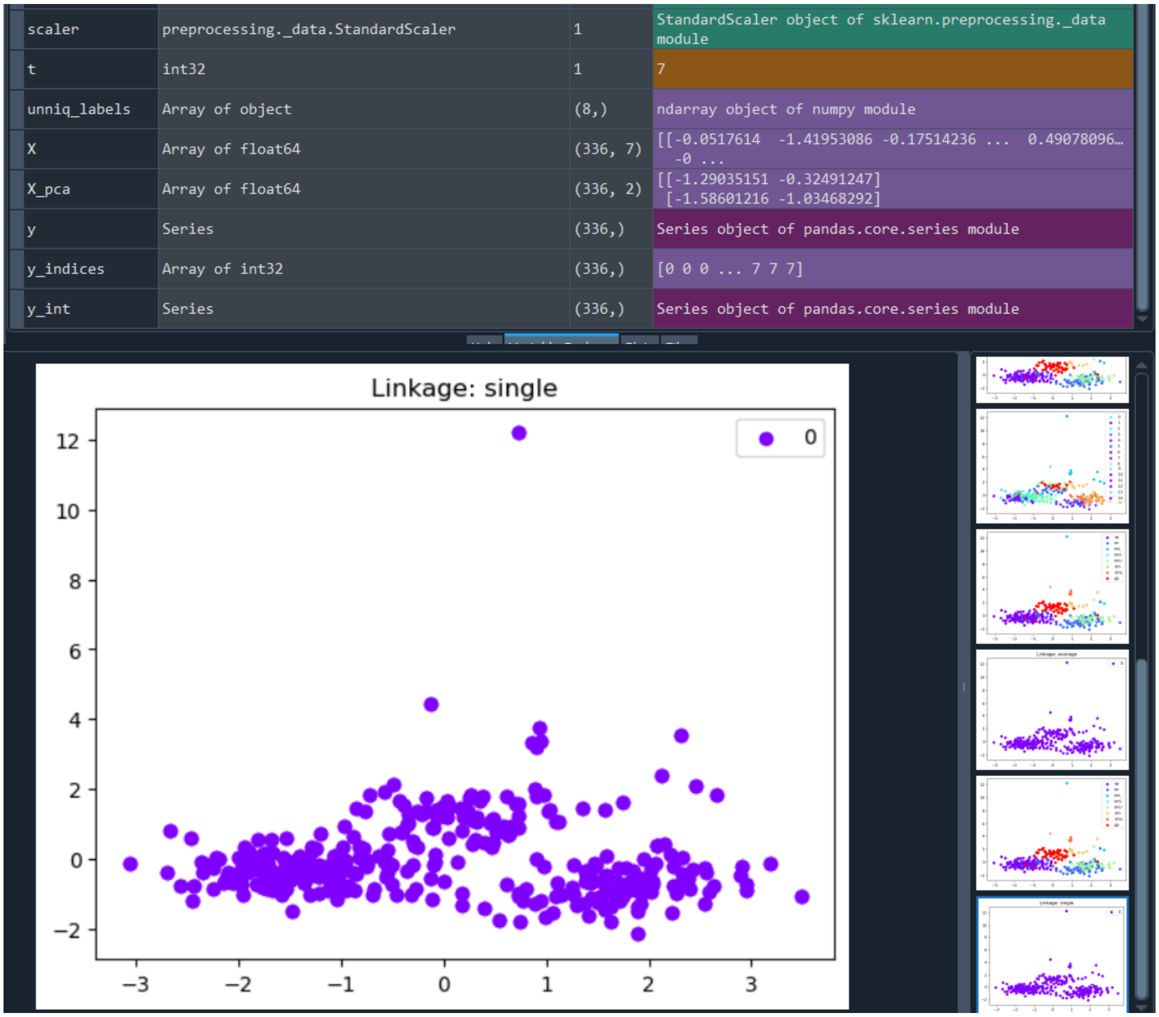
Scripts:

```
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()
```

```
In [9]: runfile('D:/AI/labwork_2/assignment2.py', wdir='D:/AI/labwork_2')
AgglomerativeClustering, linkage: single
Number of clusters: 1
Elements in each cluster: [336]
Cluster stats:
 [[143]
 [ 77]
 [  2]
 [  2]
 [ 35]
 [ 20]
 [  5]
 [ 52]]
```

Name	Type	Size	Value
c	int64	1	0
clustering	cluster._agglomerative.AgglomerativeClustering	1	AgglomerativeClustering object of sklearn.cluster._agglomerative modul ...
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int64	(336,)	[0 0 0 ... 0 0 0]
colors	Array of float64	(1, 4)	[[0.5 0. 1. 1.]]
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	335
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp':7}
nclusters	int	1	1
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 1)	[[143] [77]]
Ns	Array of int32	(1,)	[336]
p	int64	1	0
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module



scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module
t	int32	1	7
unniq_labels	Array of object	(8,)	ndarray object of numpy module
X	Array of float64	(336, 7)	[[-0.0517614 -1.41953086 -0.17514236 ... 0.49078096… -0 ...
X_pca	Array of float64	(336, 2)	[[-1.29035151 -0.32491247] [-1.58601216 -1.03468292]
y	Series	(336,)	Series object of pandas.core.series module
y_indices	Array of int32	(336,)	[0 0 0 ... 7 7 7]
y_int	Series	(336,)	Series object of pandas.core.series module

Agglomerative (linkage = 'single')

Scripts:

print('AgglomerativeClustering, linkage: complete')

clustering = AgglomerativeClustering(linkage='complete',n_clusters=None, distance_threshold=65)

clustering.fit(X)

clusters = clustering.labels_

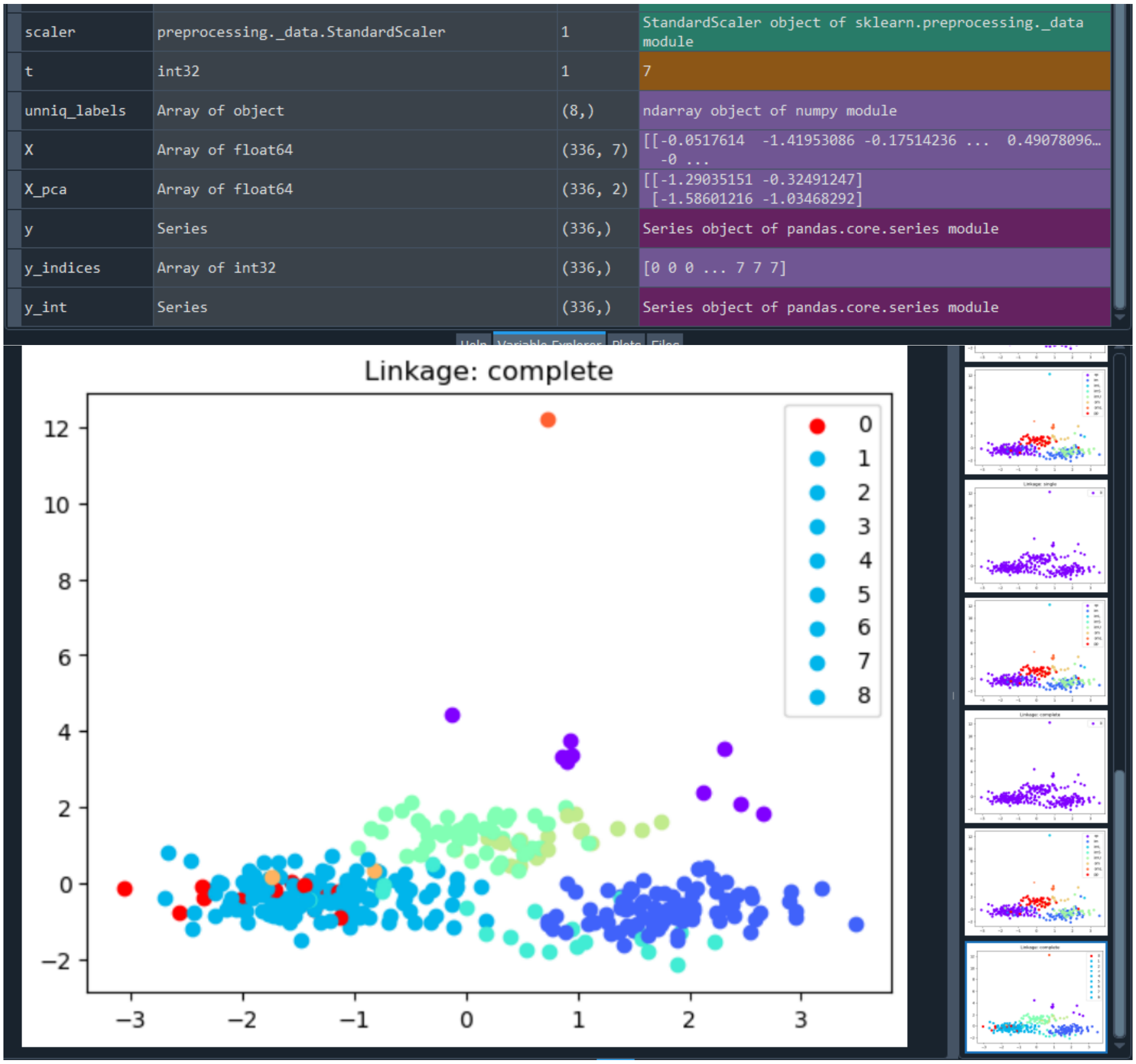
nclusters = len(np.unique(clusters))

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()
```

```
In [15]: runfile('D:/AI/labwork_2/assignment2.py', wdir='D:/AI/labwork_2')
AgglomerativeClustering, linkage: complete
Number of clusters: 9
Elements in each cluster: [  9  87 135  20  49  20   2   1  13]
Cluster stats:
 [[  0   0 128   0   2   0   0   0  13]
 [  1  52   3  20   0   0   1   0   0]
 [  1   0   0   0   0   0   0   1   0]
 [  0   1   0   0   1   0   0   0   0]
 [  1  33   0   0   0   0   1   0   0]
 [  1   0   0   0   2  17   0   0   0]
 [  5   0   0   0   0   0   0   0   0]
 [  0   1   4   0  44   3   0   0   0]]
```

Name	Type	Size	Value
c	int64	1	4
clustering	cluster._agglomerative.AgglomerativeClustering	1	AgglomerativeClustering object of sklearn.cluster._agglomerative modul ...
clustering1	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
clusters	Array of int64	(336,)	[8 2 2 ... 4 4 4]
colors	Array of float64	(9, 4)	[[5.00000000e-01 0.00000000e+00 1.00000000e+00 1.0000… [2.490 ...
df	DataFrame	(336, 9)	Column names: 0, 1, 2, 3, 4, 5, 6, 7, 8
i	int	1	335
idxs	Array of int64	(52,)	[284 285 286 ... 333 334 335]
l	str	2	pp
label_to_index	dict	8	{'cp':0, 'im':1, 'imL':2, 'imS':3, 'imU':4, 'om':5, 'omL':6, 'pp':7}
nclusters	int	1	9
Ncolors	int	1	8
nlabels	int	1	8
NNs	Array of int32	(8, 9)	[[0 0 128 ... 0 0 13] [1 52 3 ... 1 0 0]
Ns	Array of int32	(9,)	[9 87 135 20 49 20 2 1 13]
p	int64	1	4
pca	decomposition._pca.PCA	1	PCA object of sklearn.decomposition._pca module



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.