# oct-31

October 31, 2025

[1]:
```
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

[2]:
```
iris = load_iris()
```

[3]:
```
X = iris.data
```

[4]:
```
feature_names = iris.feature_names
```

[5]:
```
df = pd.DataFrame(X, columns=feature_names)
```

[6]:
```
df.shape
```

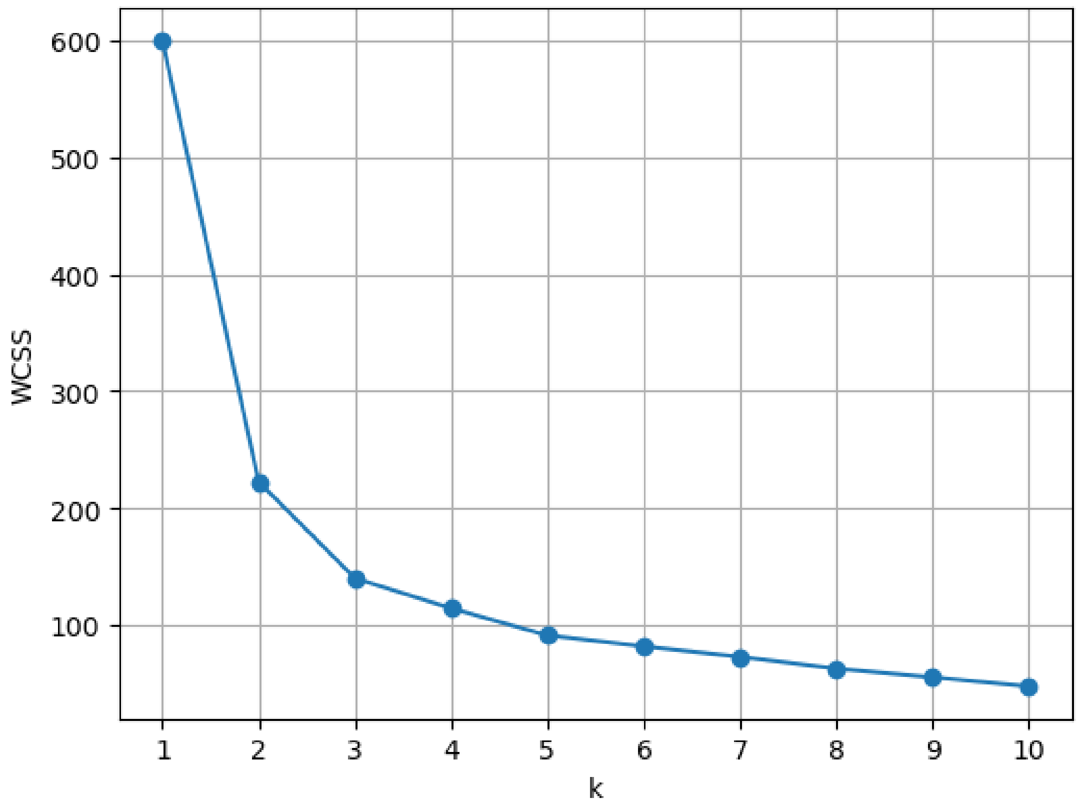
[6]:
```
(150, 4)
```

[7]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[8]:
```
### Elbow Method to find K
import matplotlib.pyplot as plt
wcss = []
ks = range(1,11)
```

[9]:
```
for k in ks:
    km = KMeans(n_clusters=k, n_init = 10, random_state = 42)
    km.fit(X_scaled)
    wcss.append(km.inertia_)
```

[10]:
```
plt.figure()
plt.plot(ks,wcss, marker="o")
plt.xlabel("k")
plt.ylabel("WCSS")
plt.xticks(list(ks))
plt.grid()
plt.show()
```



```
[11]: import numpy as np
```

```
[12]: K = 3
```

```
[13]: kmeans = KMeans(n_clusters = K, init="k-means++", n_init = 10, random_state =␣
        ↪42)
```

```
[14]: labels = kmeans.fit_predict(X_scaled)
```

```
[15]: print("Cluster sizes: ", np.bincount(labels))
```

```
Cluster sizes:  [53 50 47]
```

```
[16]: print("Total WCSS:", kmeans.inertia_)
```

```
Total WCSS: 139.8204963597498
```

```
[17]: print("Centroids in scaled space: ", kmeans.cluster_centers_)
```

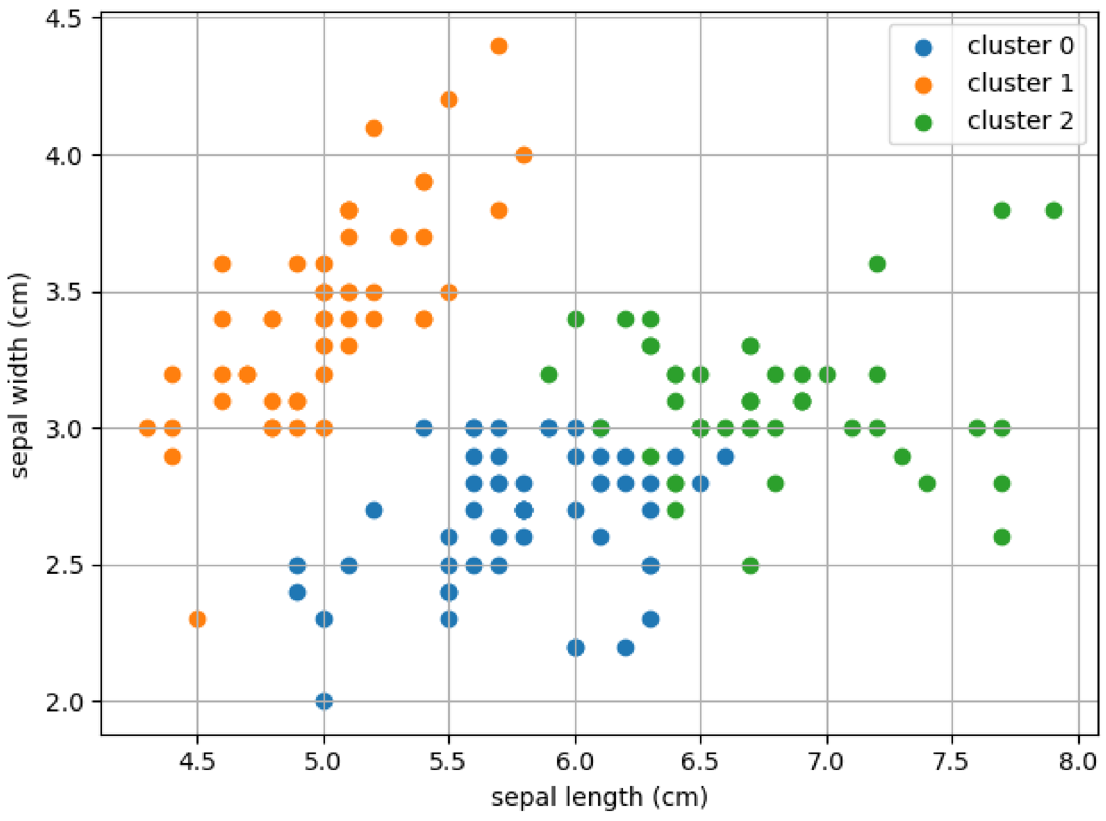
```
Centroids in scaled space:  [[-0.05021989 -0.88337647  0.34773781  0.2815273 ]
 [-1.01457897  0.85326268 -1.30498732 -1.25489349]
 [ 1.13597027  0.08842168  0.99615451  1.01752612]]
```

[18]:
```
feature_names
```

[18]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

[19]:
```
i, j = 0, 1
```

[20]:
```
plt.figure()
for c in range(K):
    mask = (labels == c)
    plt.scatter(X[mask, i], X[mask, j], label=f"cluster {c}")
    plt.xlabel(feature_names[i])
    plt.ylabel(feature_names[j])
    plt.grid()
    plt.tight_layout()
    plt.legend()
plt.show()
```



[21]:
```
new_samples = np.array([
    [5,3.5,1.3,0.3],
    [6,2.7,5.1,1.6]
])
```

[22]:
```
new_scaled = scaler.transform(new_samples)
```

[23]:
```
pred = kmeans.predict(new_scaled)
```

[24]:
```
pred
```

[24]:
```
array([1, 0], dtype=int32)
```

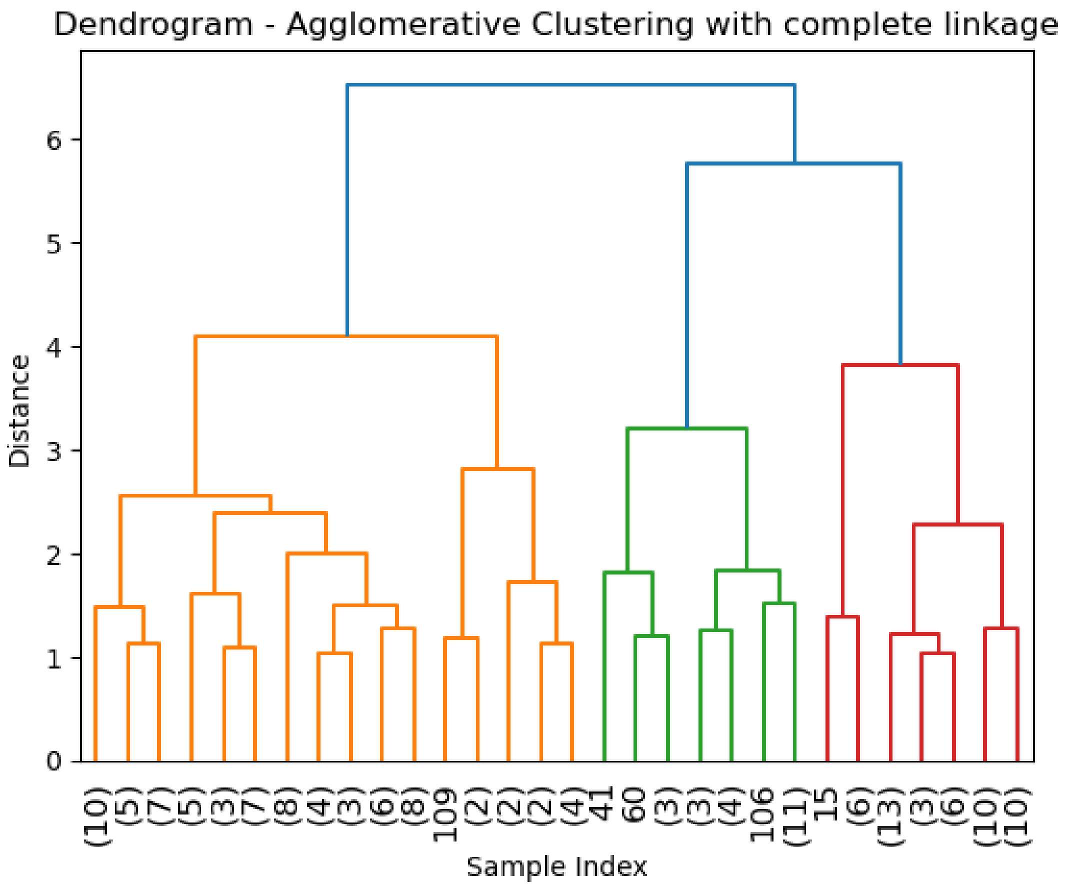
[ ]:
```

```

[25]:
```
from sklearn.cluster import AgglomerativeClustering
```

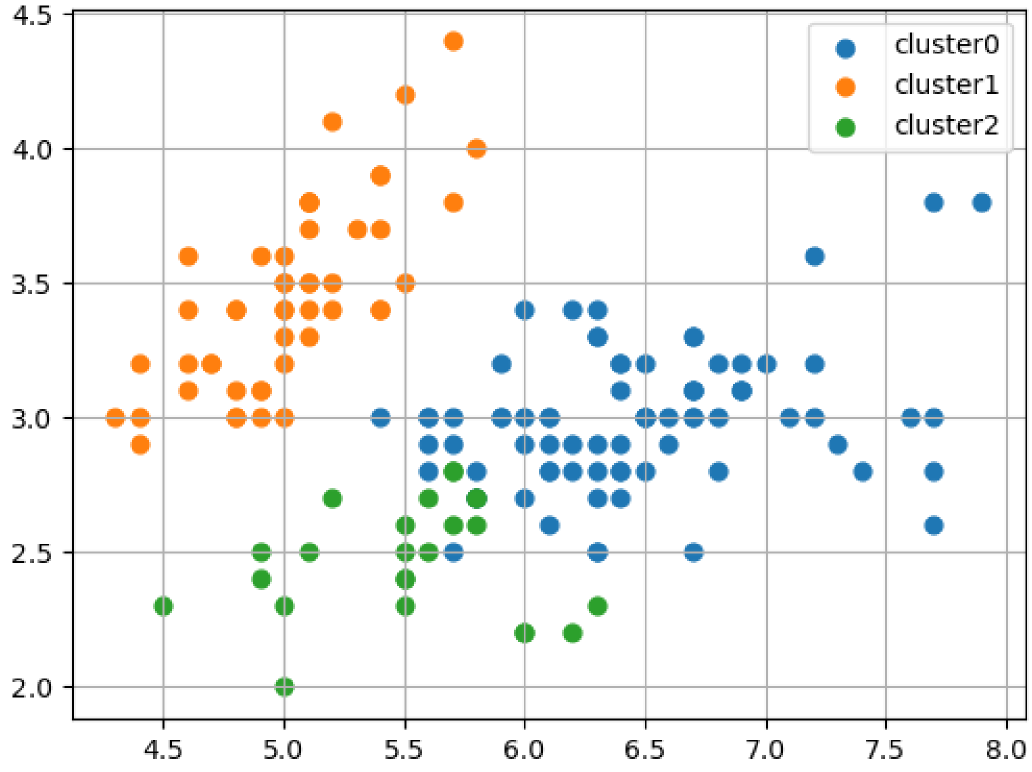
[28]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
```

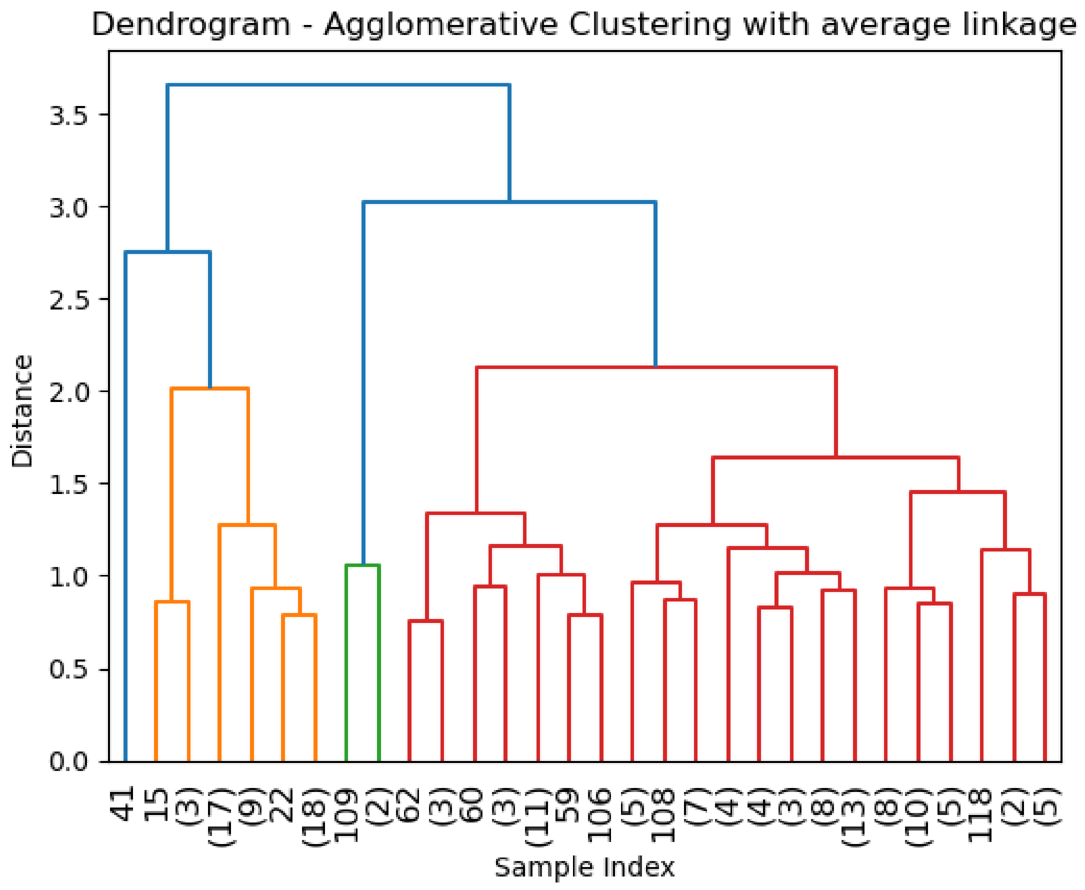
[55]:
```
linkage_methods = ["complete", "average", "single", "ward"]
```

[64]:
```
#Create the dendrogram
for linkage_method in linkage_methods:

    Z = linkage(X_scaled, method = linkage_method)

    plt.figure()
    dendrogram(Z, truncate_mode='lastp', p = 30, leaf_rotation = 90,␣
 ↪leaf_font_size = 12)
    plt.xlabel("Sample Index")
    plt.ylabel("Distance")
    plt.title(f"Dendrogram - Agglomerative Clustering with {linkage_method}␣
 ↪linkage")
    plt.show()

    agg_cluster = AgglomerativeClustering(n_clusters = 3, linkage =␣
 ↪linkage_method)
    agg_labels = agg_cluster.fit_predict(X_scaled)
    print(f"Cluster Sizes:  {np.bincount(agg_labels)}")

    plt.figure()
    i, j = 0, 1
    for c in range(3):
        mask = (agg_labels == c)
        plt.scatter(X[mask, i], X[mask, j], label = f"cluster{c}", s=40)
        plt.legend()
        plt.grid()
    plt.show()
```

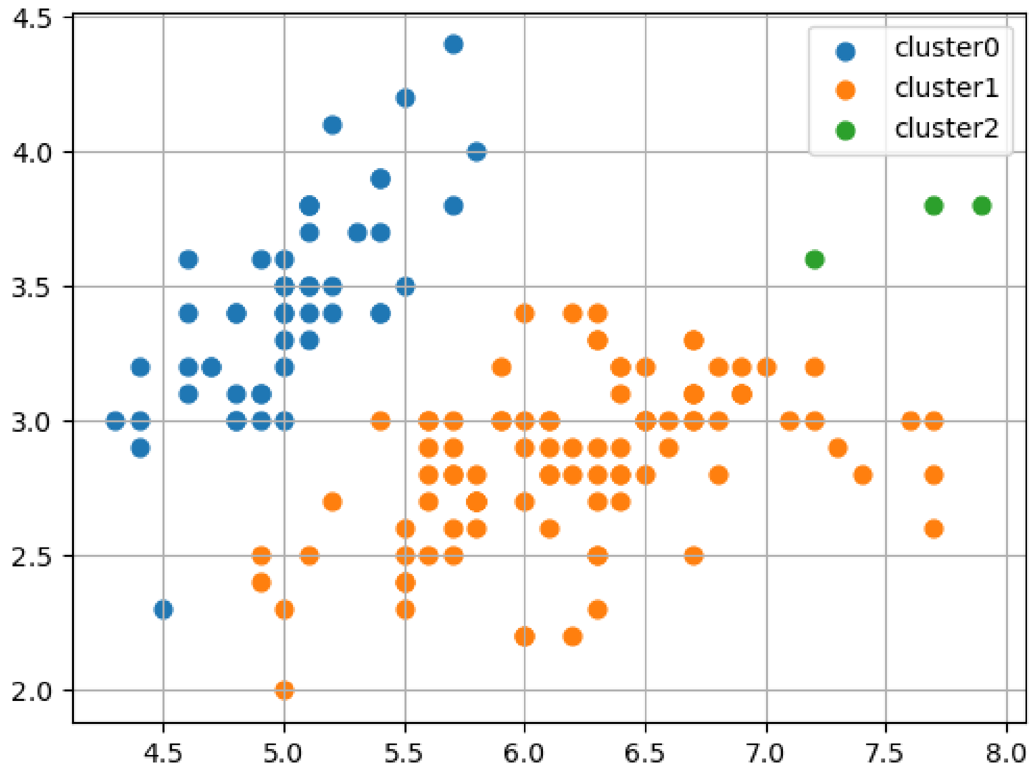


```
Cluster Sizes:  [77 49 24]
```

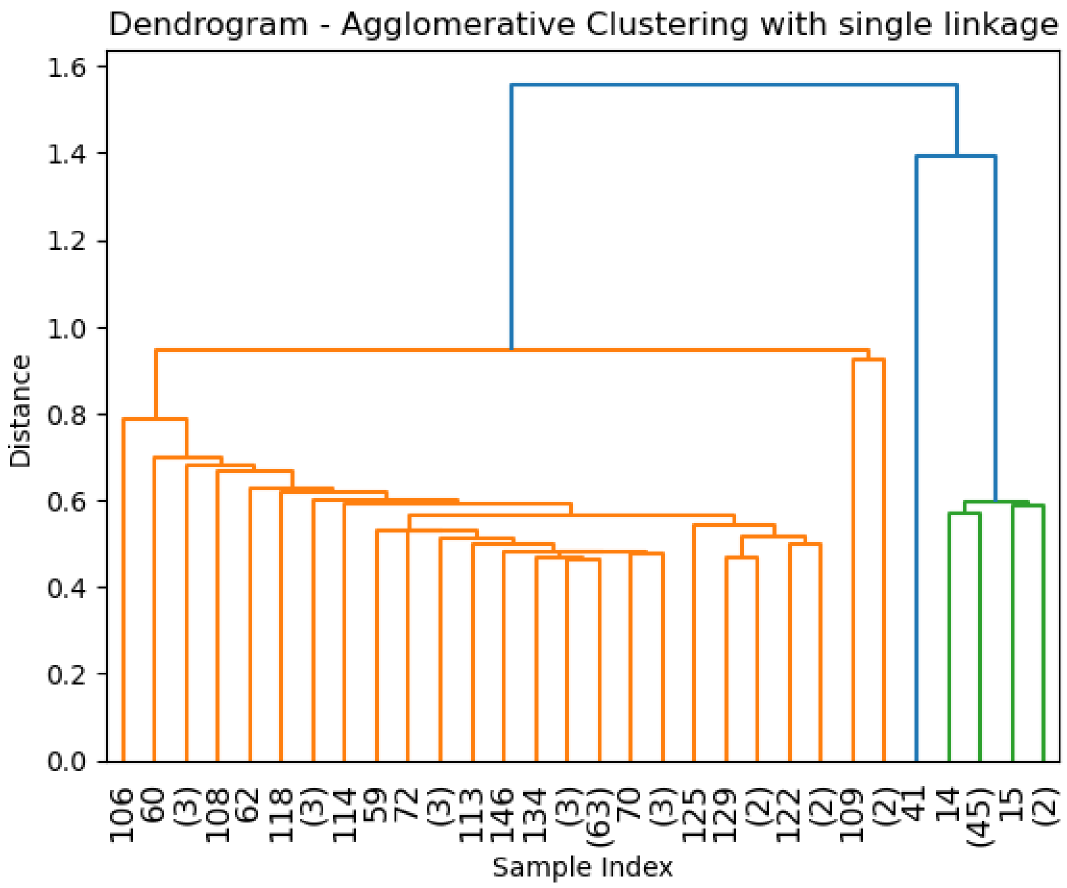

cluster0
cluster1
cluster2
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



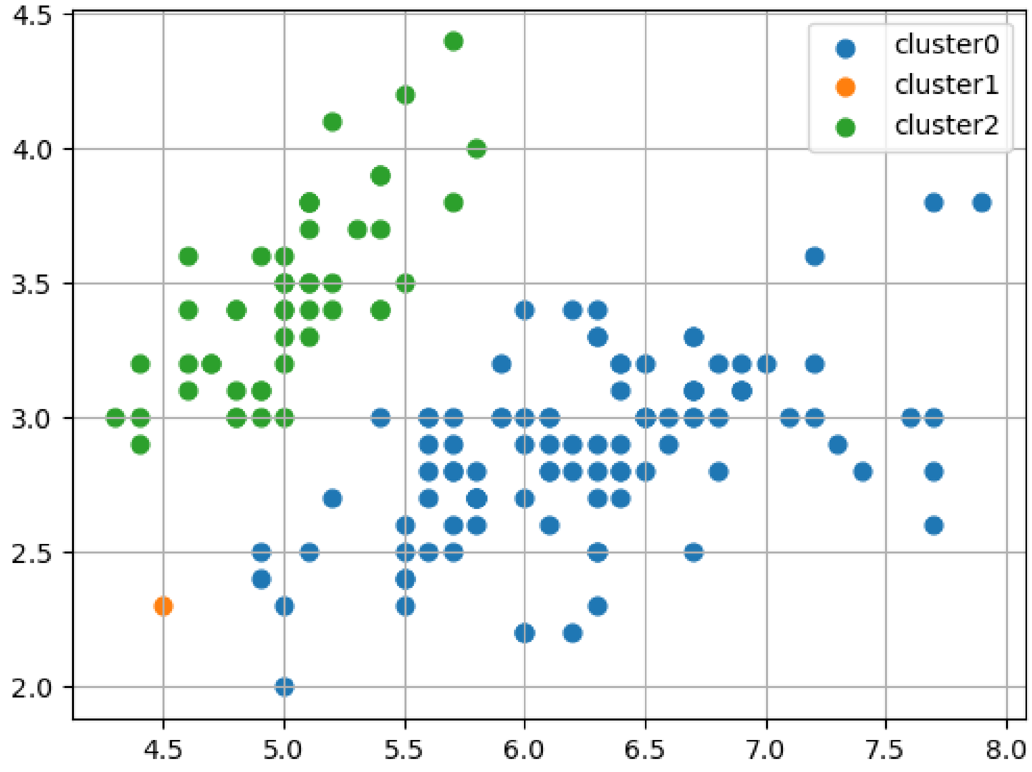
```
Cluster Sizes:  [50 97  3]
```



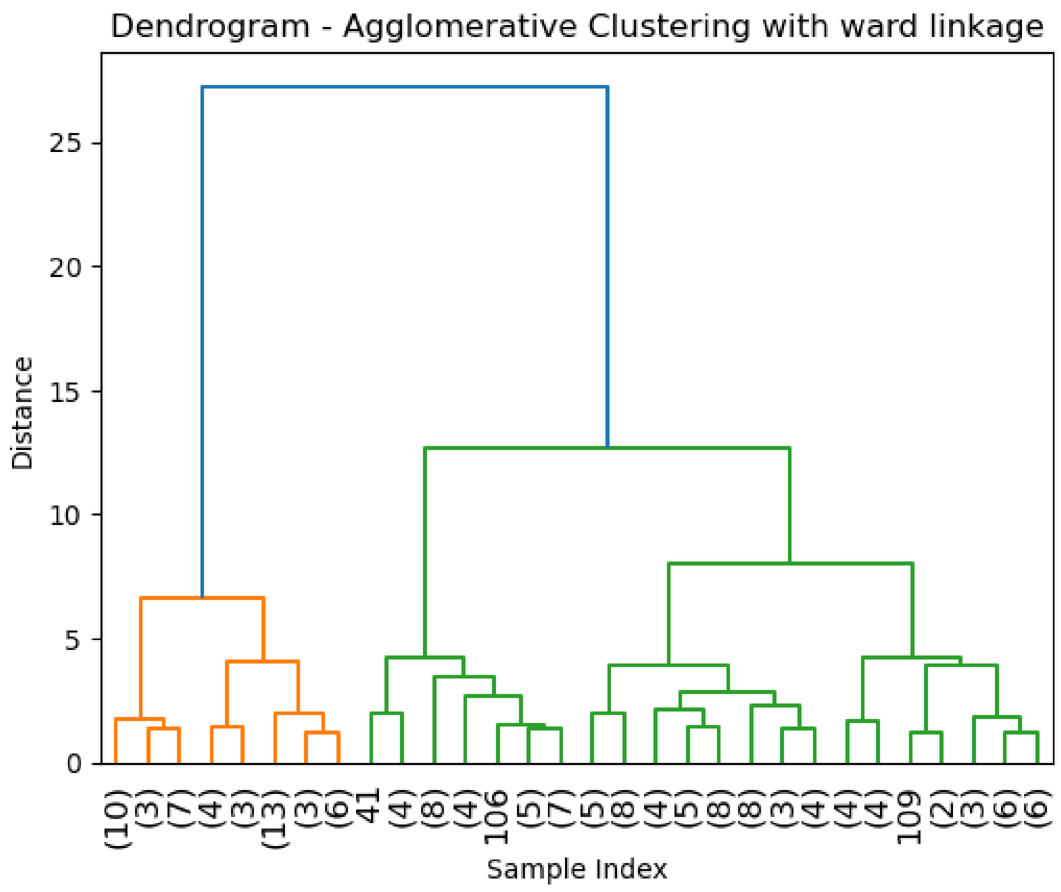
cluster0
cluster1
cluster2
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



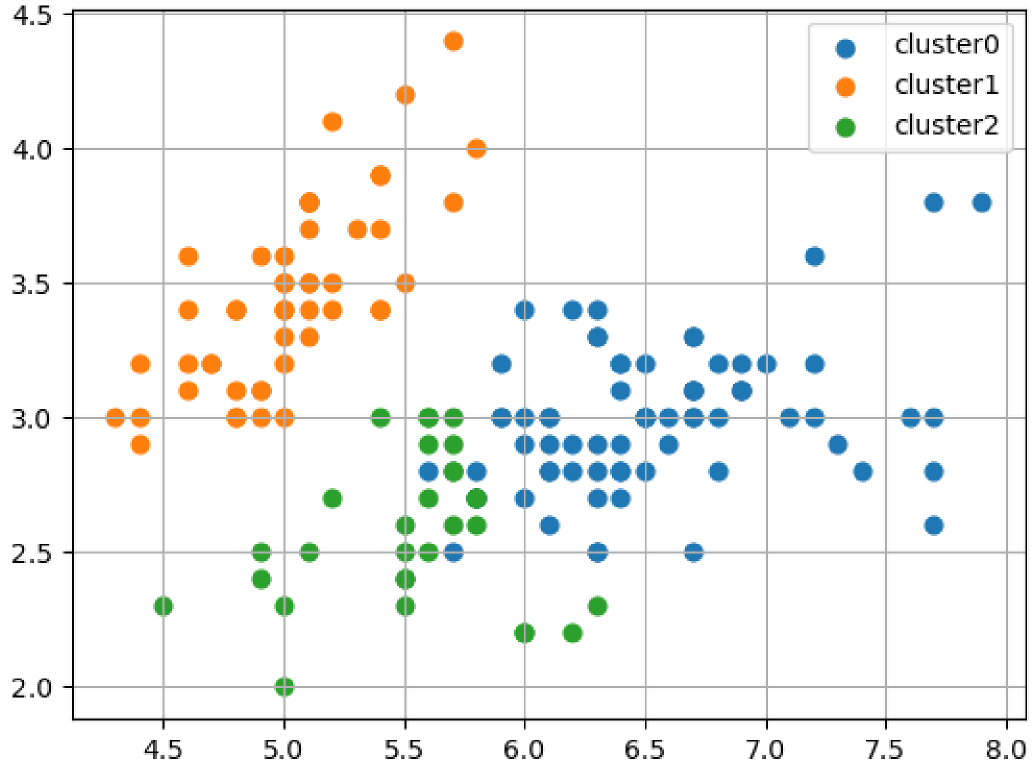
```
Cluster Sizes:  [100   1  49]
```



cluster0
cluster1
cluster2
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



```
Cluster Sizes:  [71 49 30]
```


cluster0
cluster1
cluster2
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

```
[ ]:
```