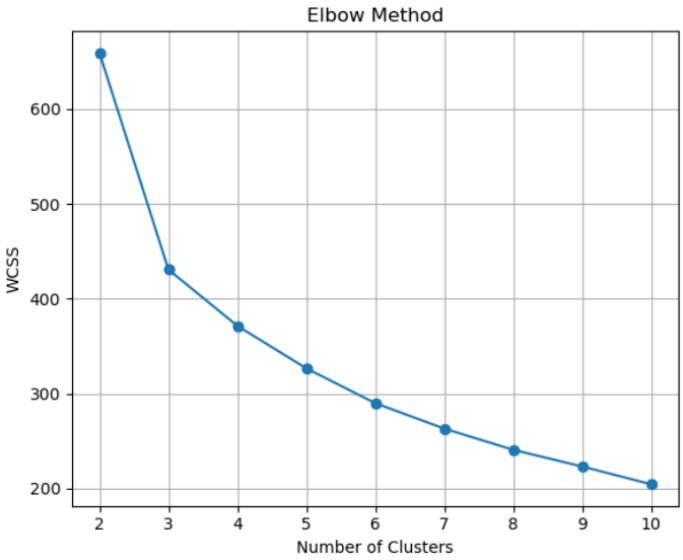
Clustering for Seeds Dataset

Find out optimum number of clusters from the Elow Method



The WCSS decreases as the number of clusters increases. However, the rate of decrease slows down at a certain point, forming an "elbow" shape.

The optimal number of clusters is usually where the **elbow or bend occurs**, indicating the point beyond which adding more clusters provides **diminishing returns** in reducing variance.

- In the given plot, there is a **sharp drop from 2 to 3 clusters**.
- The rate of decrease slows down significantly after **3rd cluster**, indicating a potential "elbow point."

I am choosing 3 clusters.

1. Do the k-means clustering for k=3, the actual number of clusters in the dataset. Evaluate the quality of clustering using the intrinsic method, Silhouette coefficient, and also the extrinsic method purity.

KMeans Clustering

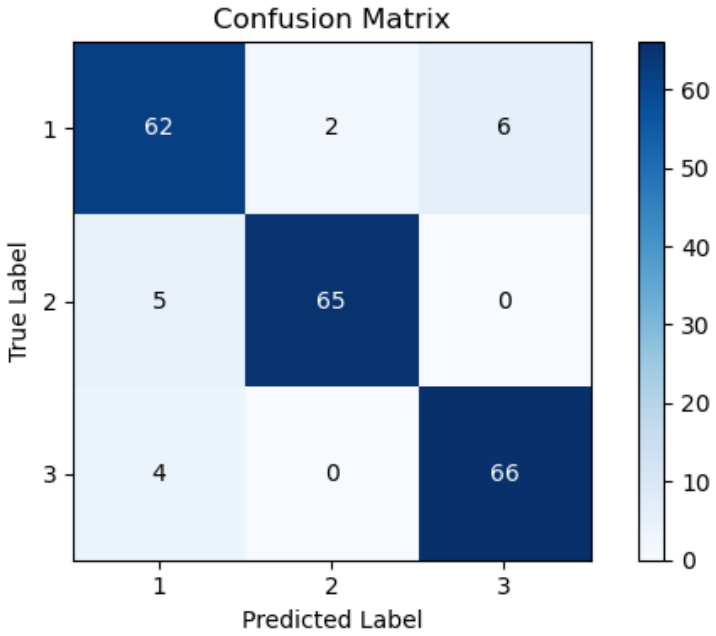
Accuracy: **0.9190**

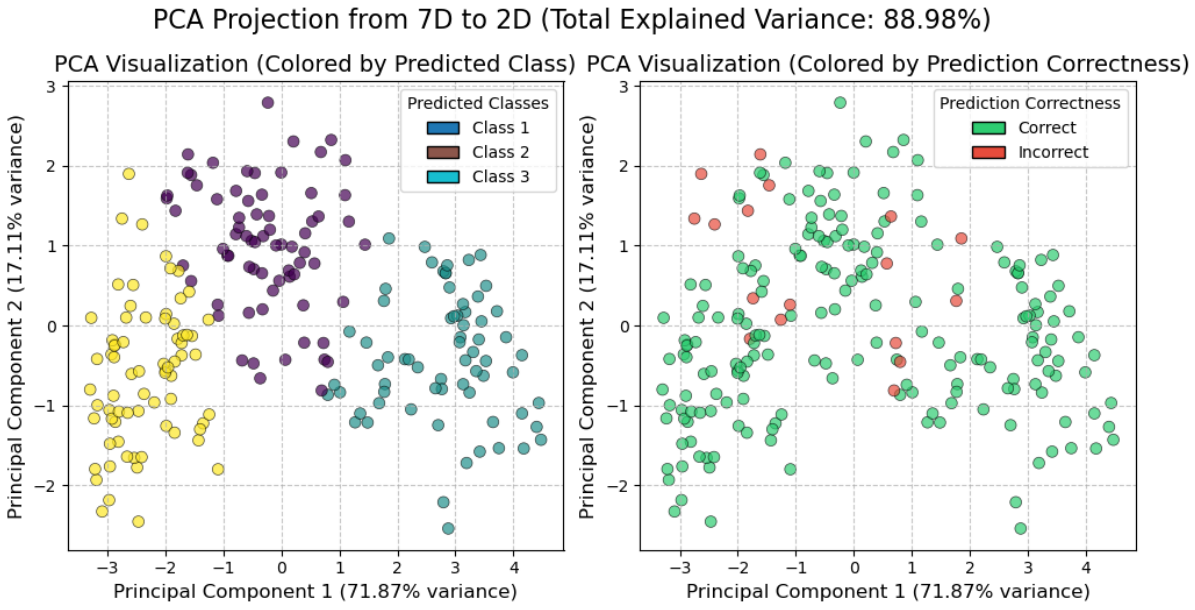
F1 Score: **0.9193**

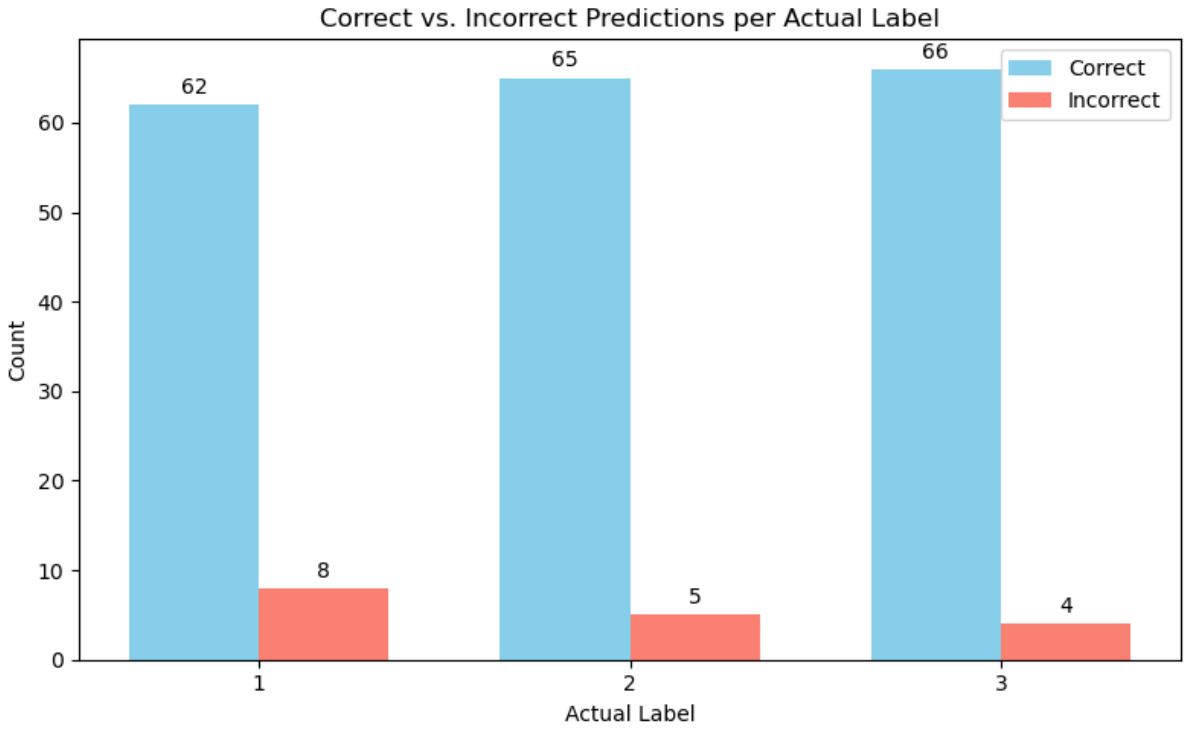
Precision: **0.9200**

Recall: **0.9190**

Purity Score: **0.9190**







Agglomerative Hierarchical Clustering with different linkage methods

(single, complete, average, and ward).

I have experimented the Agglomerative Hierarchical Clustering. This is the evaluation results for each.

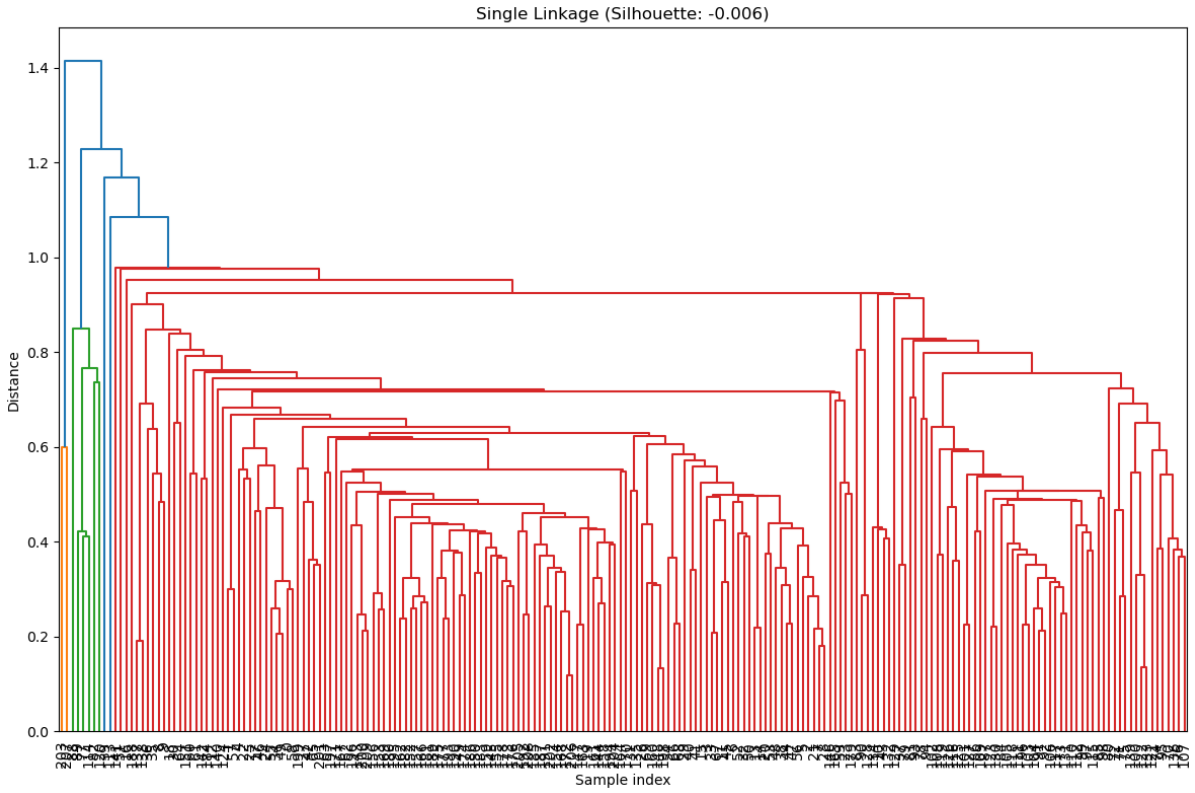
	single_Agglomerative	complete_Agglomerative	average_Agglomerative	ward_Agglomerative
Silhouette Score	-0.005642	0.350198	0.375957	0.392634
Purity Score	0.919048	0.919048	0.919048	0.919048

The results visualizations are attached to the code notebook.

Dendrogram Analysis

The goal is to select a level where the clusters are distinct (large vertical gaps) but not overly fragmented (too many small clusters).

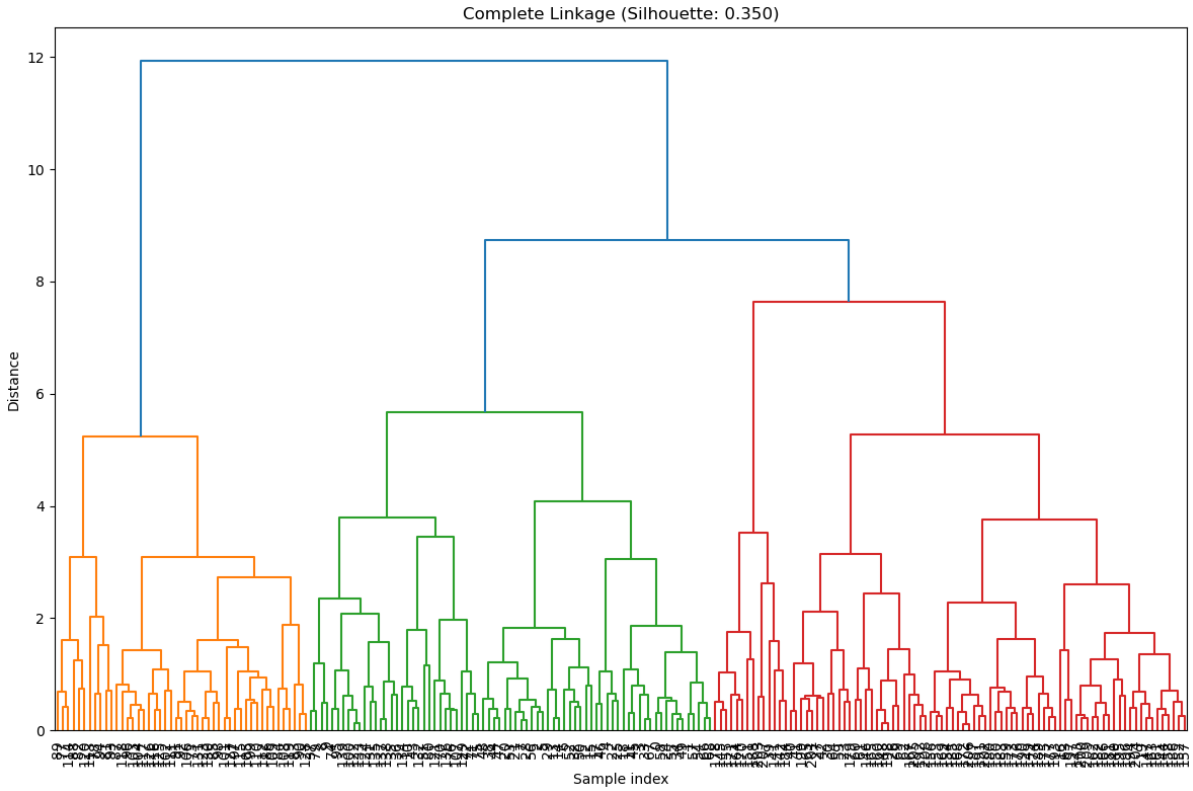
Simple Linkage



This is not a good separation cluster distances are very close.

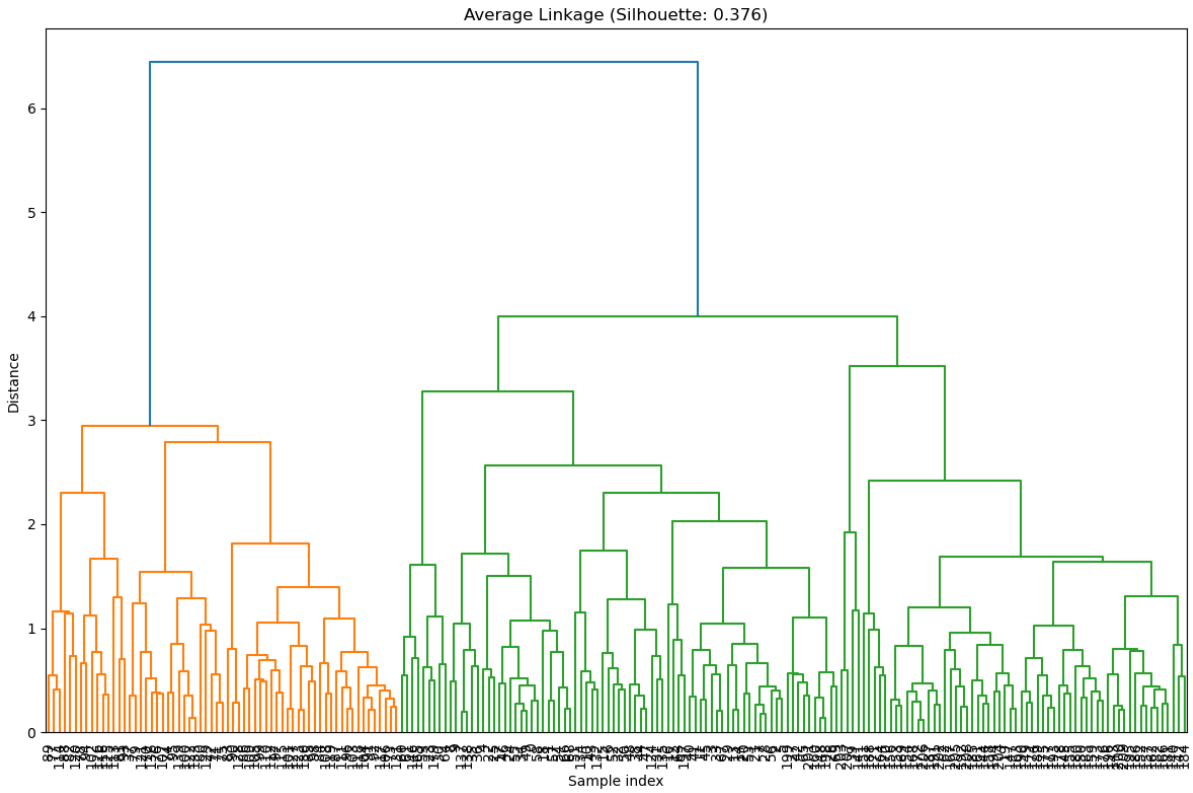
No good separation.

Complex Linkage



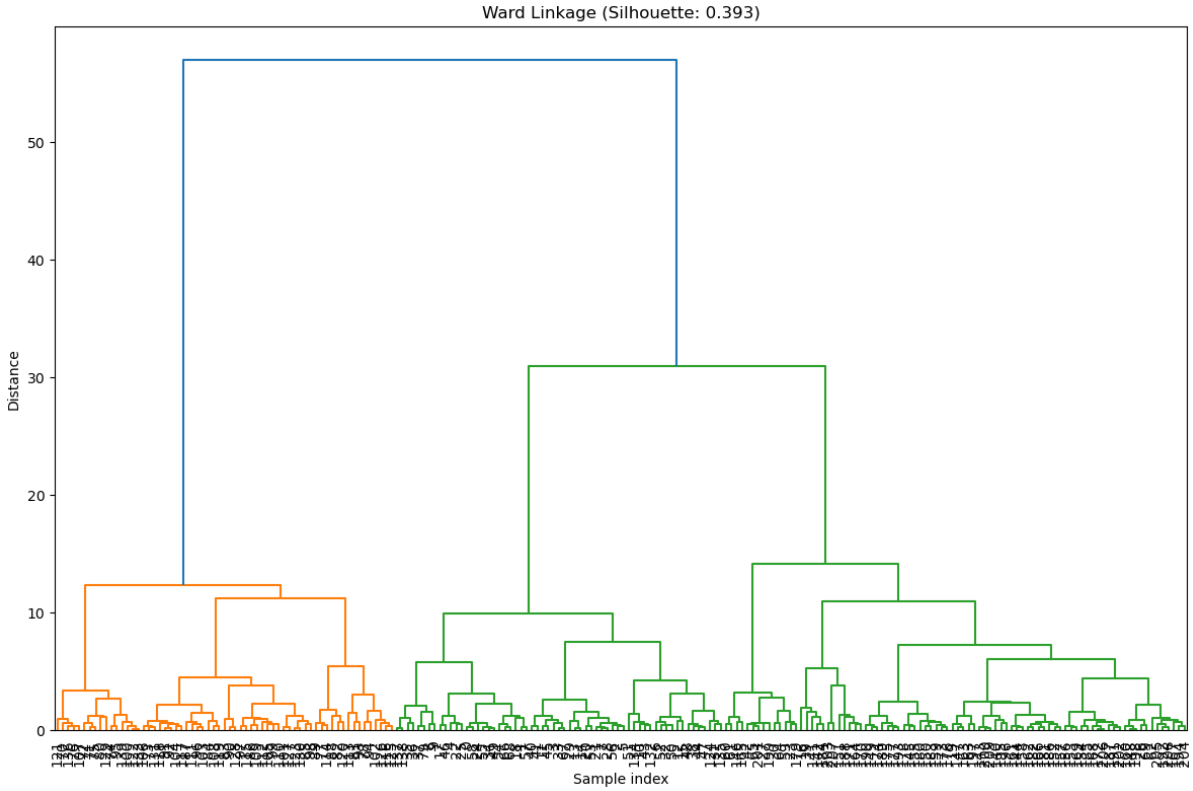
We can separate from around distance 6; creating 4 clusters.

Average Linkage



We can separate from around distance 3.5; creating 3 clusters.

Ward Linkage

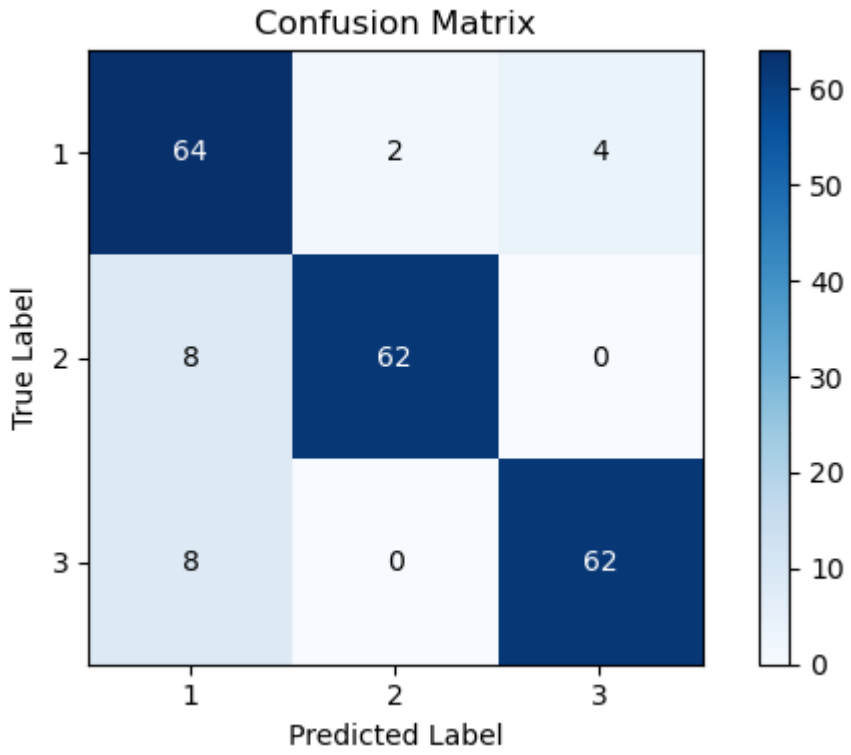


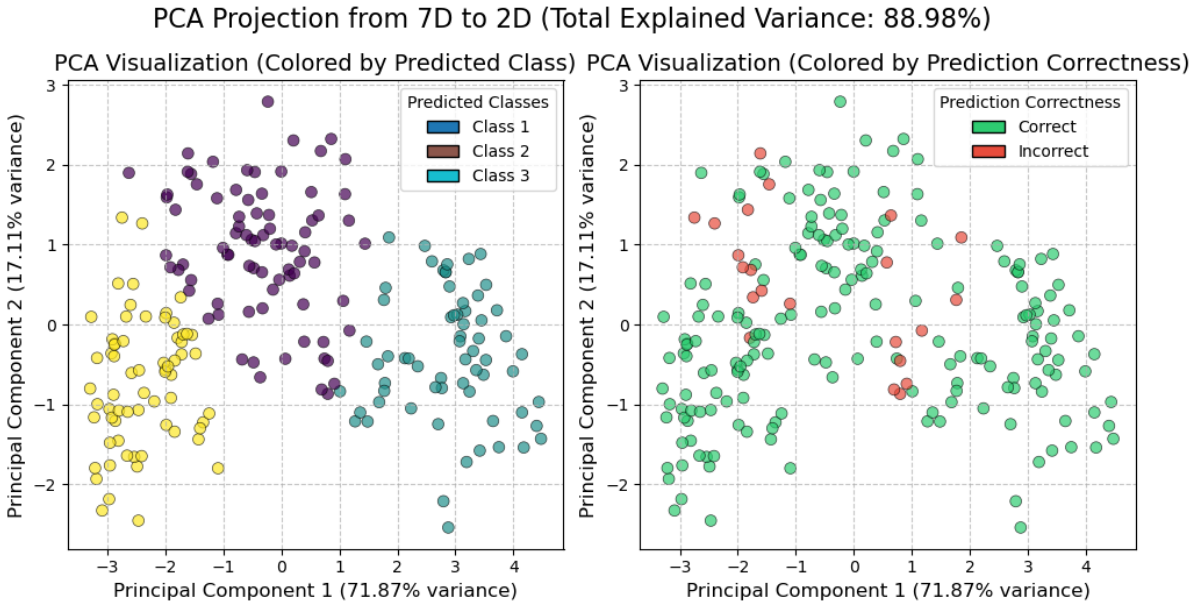
This is a comparatively good separation. Good distances within clusters.

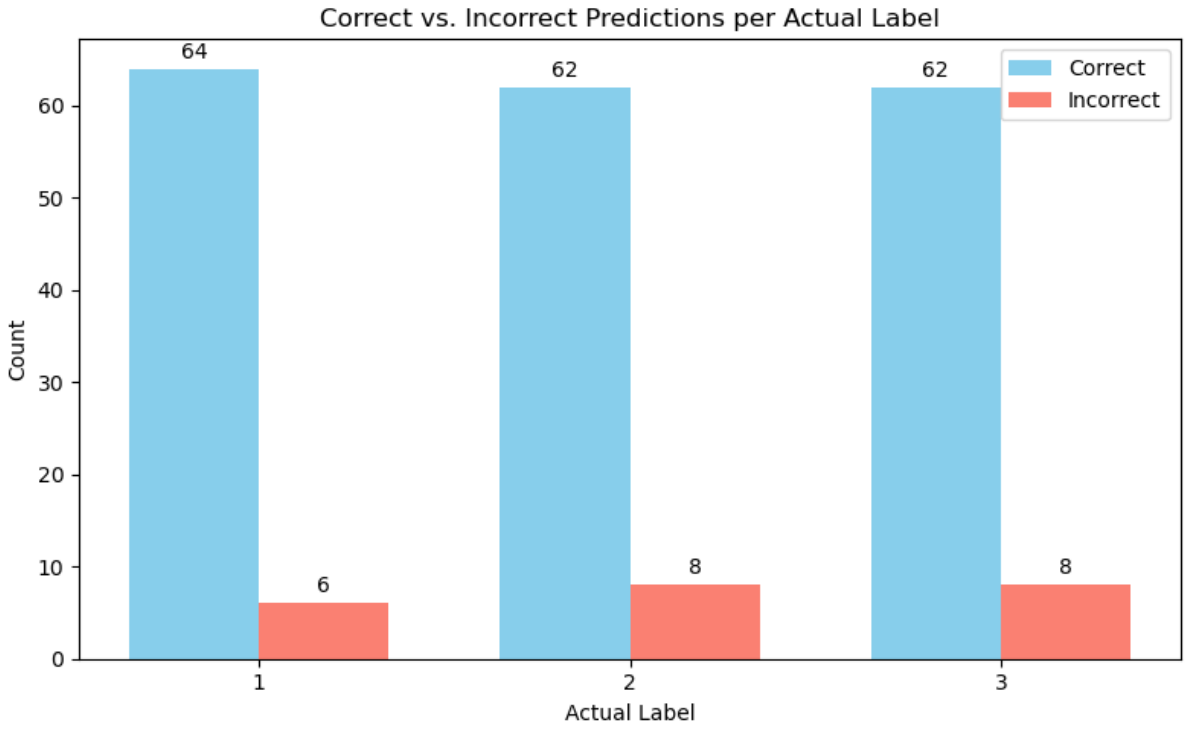
We can separate from around distance 30; creating 3 clusters.

K-Mediods algorithm

Silhouette:	0.399208
F1 Score:	0.8968
Precision:	0.9027
Recall:	0.8952
Purity Score:	0.895


Confusion Matrix
True Label
Predicted Label
1
2
3
64
2
4
8
62
0
8
0
62
60
50
40
30
20
10
0


PCA Projection from 7D to 2D (Total Explained Variance: 88.98%)
PCA Visualization (Colored by Predicted Class)
PCA Visualization (Colored by Prediction Correctness)
Predicted Classes
Class 1
Class 2
Class 3
Prediction Correctness
Correct
Incorrect
Principal Component 1 (71.87% variance)
Principal Component 2 (17.11% variance)



Limitations of k-means which are addressed by k-medoids algorithms

One big limitation of k-means is that it's super sensitive to outliers. In k-means, the algorithm tries to find the average (mean) of all the points in a cluster to decide where the centroid should be. If there's an outlier—like a point that's way far away from the others—it can pull the mean in a weird direction, messing up the whole cluster.

K-medoids, on the other hand, doesn't use the mean. Instead, it picks an actual data point (called a medoid) as the center of the cluster. Since it's choosing a real point and not calculating an average, outliers don't throw it off as much.

Another problem with k-means is that it assumes all clusters are kind of ball-shaped and roughly the same size. It uses Euclidean distance to figure out which points belong to which cluster. If data has clusters that are stretched out, oddly shaped, or different sizes, k-means struggles to get it right. K-medoids is more flexible because it can work with different ways of measuring distance, not just Euclidean, so it can handle weirder cluster shapes better.

K-means also has a hard time when the data has a lot of noise—like random points that don't really fit anywhere. Since it's always trying to minimize the distance between points and the cluster mean, noisy data can confuse it and lead to bad clusters. K-

medoids is tougher against noise because it sticks to using real data points as centers, so it's less likely to get distracted by random junk in the data.

Finally, k-means can sometimes get stuck depending on where it starts. It randomly picks starting points for the centroids, and if those guesses are bad, the clusters can end up wonky. K-medoids still picks starting points randomly, but because it's working with actual data points as medoids, it's generally less picky about the starting position and can still find decent clusters.

The final Conclusion from the above analysis

	KMeans	single_Agglomerative	complete_Agglomerative	average_Agglomerative	ward_Agglomerative	Kmedoids
Silhouette Score	0.400727	-0.005642	0.350198	0.375957	0.392634	0.399208
Purity Score	0.919048	0.919048	0.919048	0.919048	0.919048	0.895238

Silhouette Score provides insights into how well clusters are formed, while Purity Score measures how well they match ground truth labels.

The two metrics agree, on KMeans clustering algorithm.