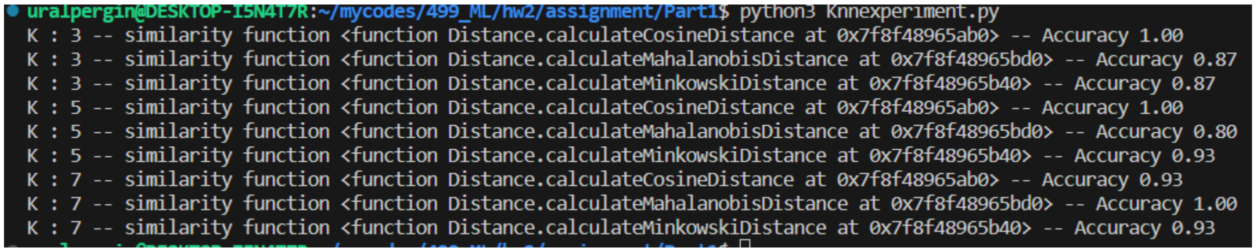
PART1)

KNNEXPERIMENT:

In the experiment I tried 6 hyperparameter configurations which are:

K values [3,5,7] and Similarity function values [Cosine, Minkowski, Mahalanobis]

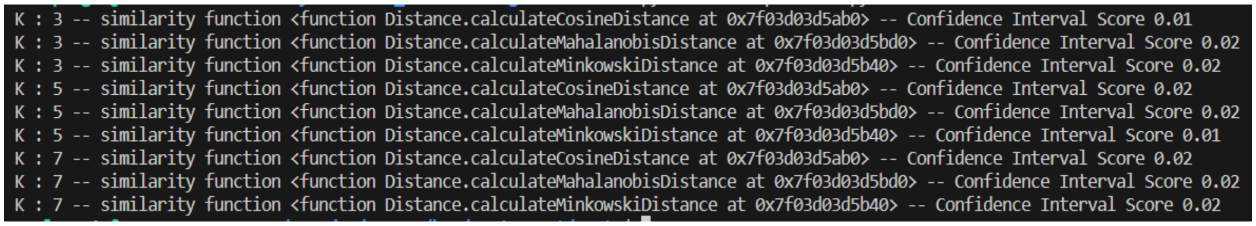
The resulting accuracy scores after the cross validation is as seen in the picture below which are very optimistic results.



```
uralpergin@DESKTOP-I5N4T7R:~/mycodes/499_ML/hw2/assignment/Part1$ python3 Knnexperiment.py
  K : 3 -- similarity function <function Distance.calculateCosineDistance at 0x7f8f48965ab0> -- Accuracy 1.00
  K : 3 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f8f48965bd0> -- Accuracy 0.87
  K : 3 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f8f48965b40> -- Accuracy 0.87
  K : 5 -- similarity function <function Distance.calculateCosineDistance at 0x7f8f48965ab0> -- Accuracy 1.00
  K : 5 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f8f48965bd0> -- Accuracy 0.80
  K : 5 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f8f48965b40> -- Accuracy 0.93
  K : 7 -- similarity function <function Distance.calculateCosineDistance at 0x7f8f48965ab0> -- Accuracy 0.93
  K : 7 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f8f48965bd0> -- Accuracy 1.00
  K : 7 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f8f48965b40> -- Accuracy 0.93
```

According to these results the best hyperparameter configuration for me is K = 7 and Function = Cosine.

I choose K=7 because it has the best overall acurracy scores compared to others. And the reason I choose Cosine is that it performs very well in all the configurations. The acurracy scores of the configurations that contains Cosine, did not vary that much compared to the other functions. It generally results in high results.



```
K : 3 -- similarity function <function Distance.calculateCosineDistance at 0x7f03d03d5ab0> -- Confidence Interval Score 0.01
K : 3 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f03d03d5bd0> -- Confidence Interval Score 0.02
K : 3 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f03d03d5b40> -- Confidence Interval Score 0.02
K : 5 -- similarity function <function Distance.calculateCosineDistance at 0x7f03d03d5ab0> -- Confidence Interval Score 0.02
K : 5 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f03d03d5bd0> -- Confidence Interval Score 0.02
K : 5 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f03d03d5b40> -- Confidence Interval Score 0.01
K : 7 -- similarity function <function Distance.calculateCosineDistance at 0x7f03d03d5ab0> -- Confidence Interval Score 0.02
K : 7 -- similarity function <function Distance.calculateMahalanobisDistance at 0x7f03d03d5bd0> -- Confidence Interval Score 0.02
K : 7 -- similarity function <function Distance.calculateMinkowskiDistance at 0x7f03d03d5b40> -- Confidence Interval Score 0.02
```

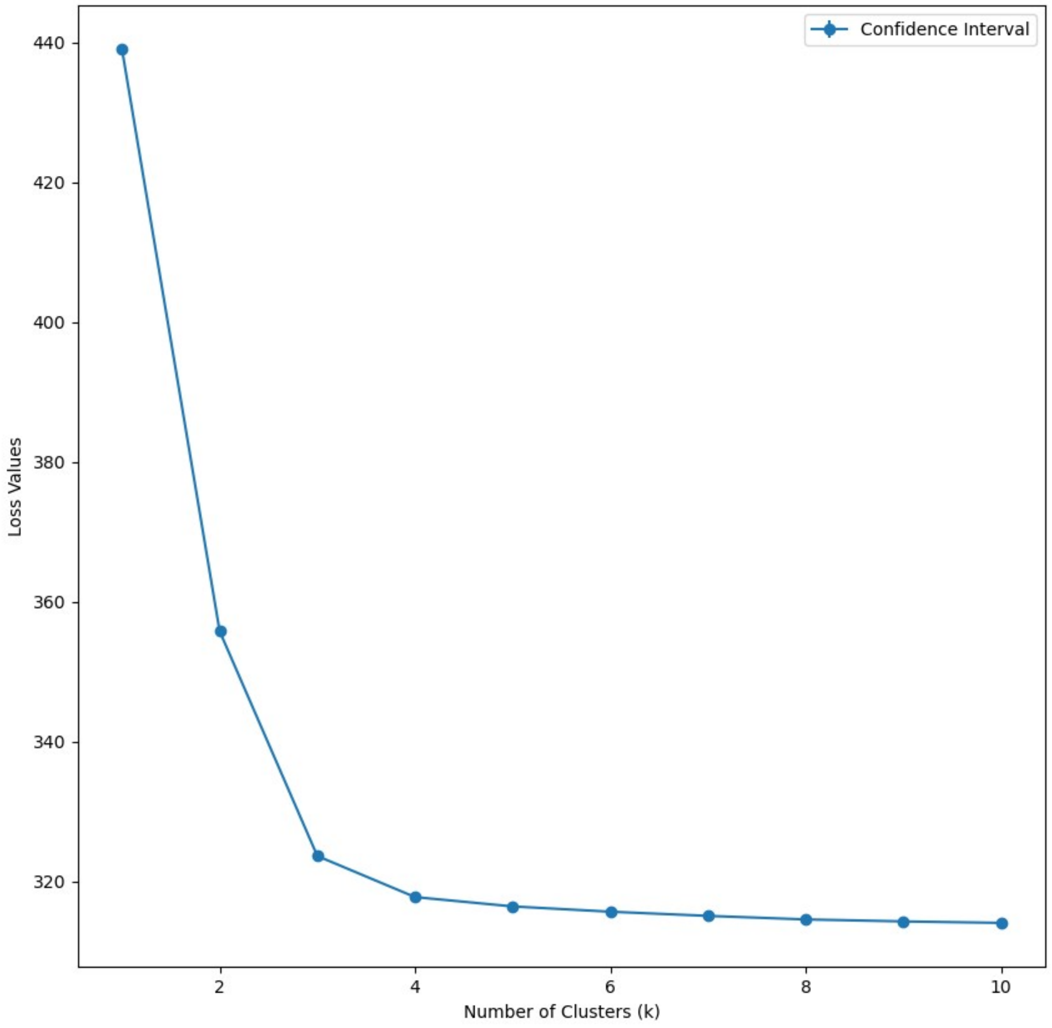
The confidence interval scores are shown in the above picture. There is a point I want to mention about this experiment. At first I thought there was a mistake in my implementation because I always get same accuracy scores (0.87, 0.93, 0.80, 1.00). Therefore I tested my cross-validation structure with sklearn.KNN classifier to see if the results are different. I saw that both KNN and my implementation of knn results in same accuracy scores. I think this is because of the dataset being very small and I wanted to mention this ambiguity in the accuracy scores.

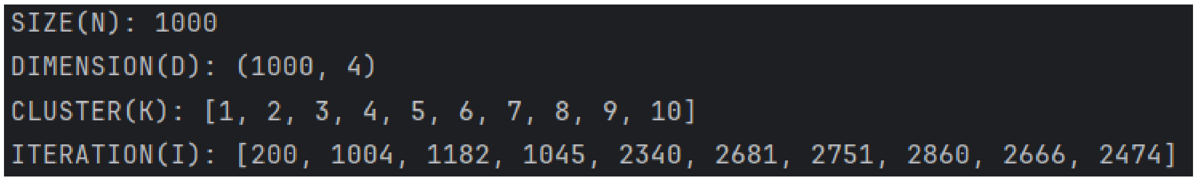
PART2)

KMEANSEXPERIMENT:

I choose k values from 1 to 10 and conducted the experiment. Confidence intervals can be seen as the error bars in the plots. Note that some of the intervals are so small that they can not be seen.

DATASET1:





```
SIZE(N): 1000
DIMENSION(D): (1000, 4)
CLUSTER(K): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ITERATION(I): [200, 1004, 1182, 1045, 2340, 2681, 2751, 2860, 2666, 2474]
```

T

CONFIDENCE INTERVALS [**K=1** 0. 0

K=2: 3.523191602386091e-14,

K=3 4.7268575524150255e-14,

K=4 6.102346860132716e-14,

K=5 7.307702058155465e-06,

K=6 0.008857633063063442,

K=7 0.013997696033352077,

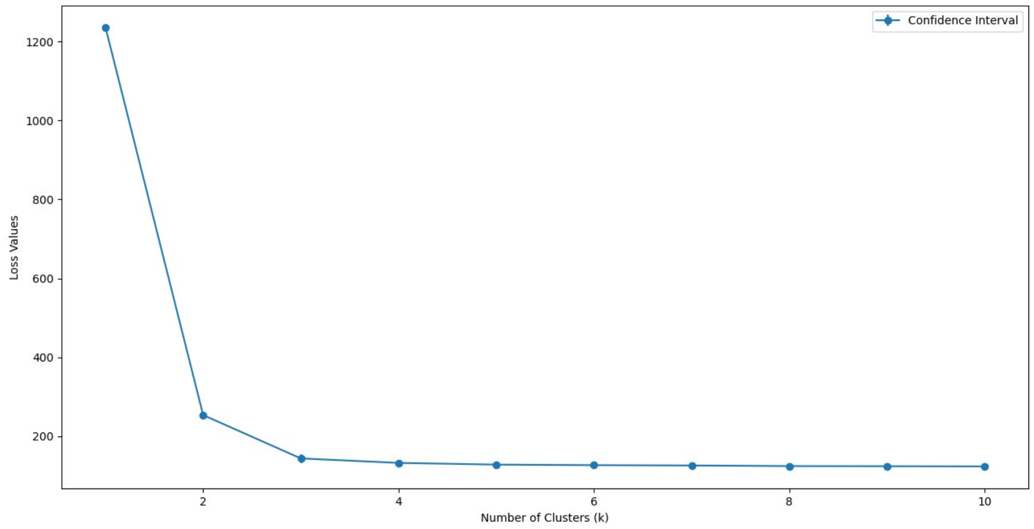
K=8 0.05377304114269558,

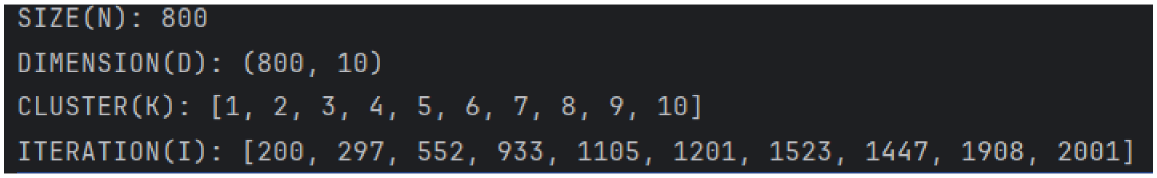
K=9 0.00816354538440198,

K=10 0.003136690520495986]

In the first dataset elbow method shows that K=3 is the most suitable option to choose since the loss values starts decreasing in a linear fashion after K=3.

DATASET2:





```
SIZE(N): 800
DIMENSION(D): (800, 10)
CLUSTER(K): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ITERATION(I): [200, 297, 552, 933, 1105, 1201, 1523, 1447, 1908, 2001]
```

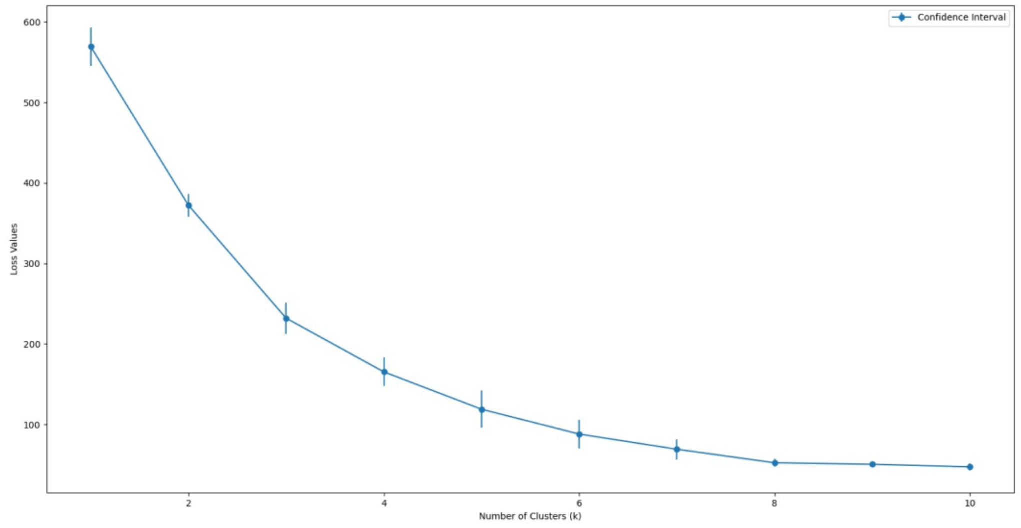
CONFIDENCE INTERVALS [0.0, 1.7615958011930455e-14, 5.372145179633831e-14, 1.4440788355477033e-13, 0.01578851232472414, 0.07326271149637134, 0.35161197121131244, 0.03821254047415171, 0.044369220238663236, 0.059816084380107305]

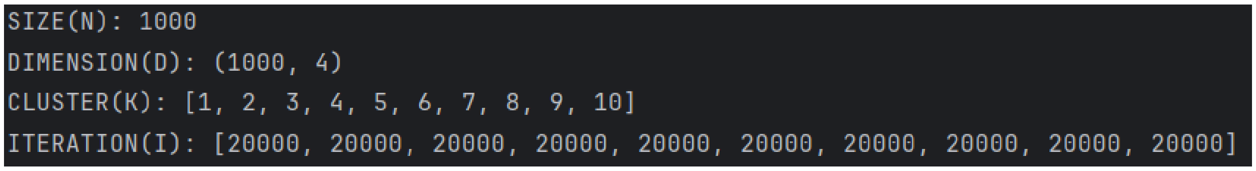
In the second dataset elbow method shows that K=2 is the most suitable option to choose since the loss values starts decreasing in a linear fashion after K=2.

KMEDOIDSEXPERIMENT:

I choose k values from 1 to 10 and conducted the experiment. Confidence intervals can be seen as the error bars in the plots. Note that some of the intervals are so small that they can not be seen.

DATASET1:



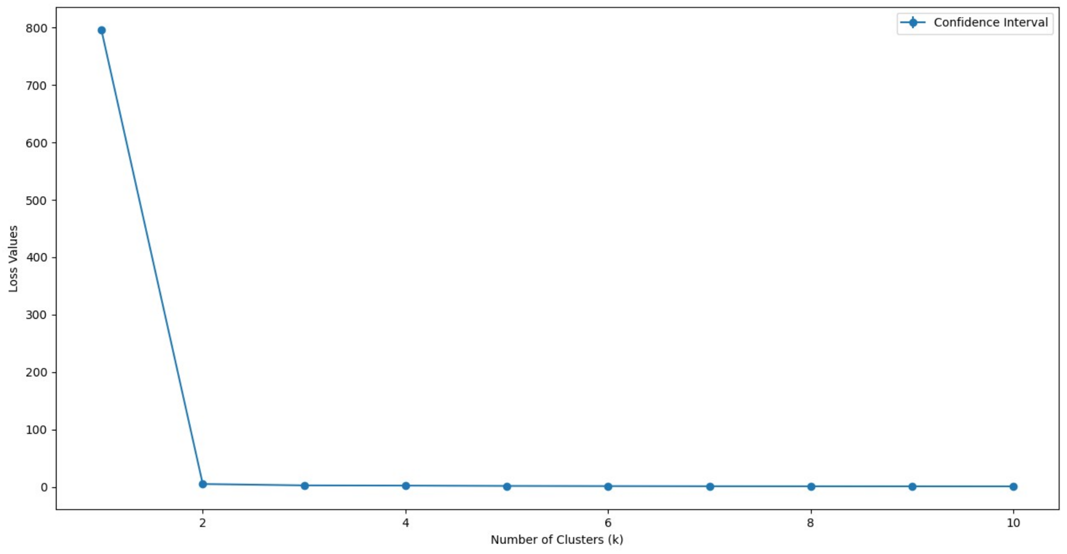


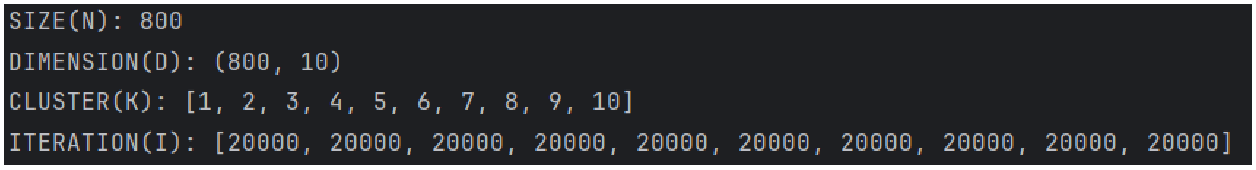
```
SIZE(N): 1000
DIMENSION(D): (1000, 4)
CLUSTER(K): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ITERATION(I): [20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000]
```

CONFIDENCE INTERVALS [23.78178760707671, 14.189143247653297, 19.15431331181971, 17.792645166850924, 23.144828961450113, 17.52759279841483, 12.756750561359247, 4.790435361842196, 3.0416158613224202, 4.517441596842708]

In the first dataset elbow method shows that K=3 is the most suitable option to choose since the loss values starts decreasing in a linear fashion after K=3. In this figure selecting the best K value seems a little bit hard for me but anyway I find K = 3 to be the optimal k value in this experiment.

DATASET2:





```
SIZE(N): 800
DIMENSION(D): (800, 10)
CLUSTER(K): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ITERATION(I): [20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000, 20000]
```

CONFIDENCE INTERVALS [0.26363543888297913, 0.2687286008817738, 0.42557535502294463, 0.2648113182415344, 0.16237300759859938, 0.07863916118089403, 0.05310184502735015, 0.07972311697929507, 0.05586478725334247, 0.02873030230849371]

In the second dataset elbow method shows that K=2 is the most suitable option to choose since the loss value almost becomes 0 after K=2.

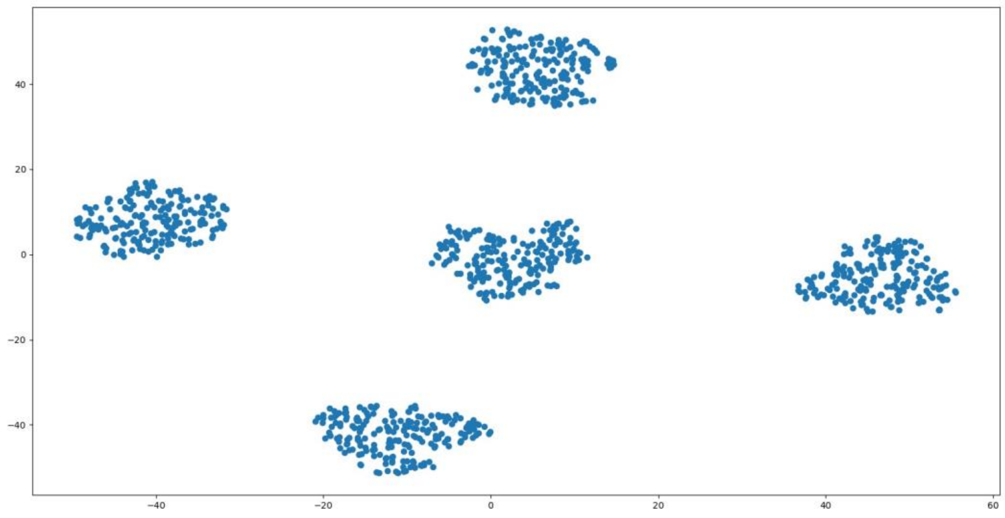
Dimensionality Reduction:

The results of the dimensionality reduction methods are different from my implementations results. Due to dimensionality reduction methods: dataset1(K = 5), dataset2(K = 4) which are different from my results in section above. The best method I found is the UMAP method which shows a clear clustering of the dataset and very distinct cluster(they behave like points and not seperated in itself), which I think is a plus compared to others.

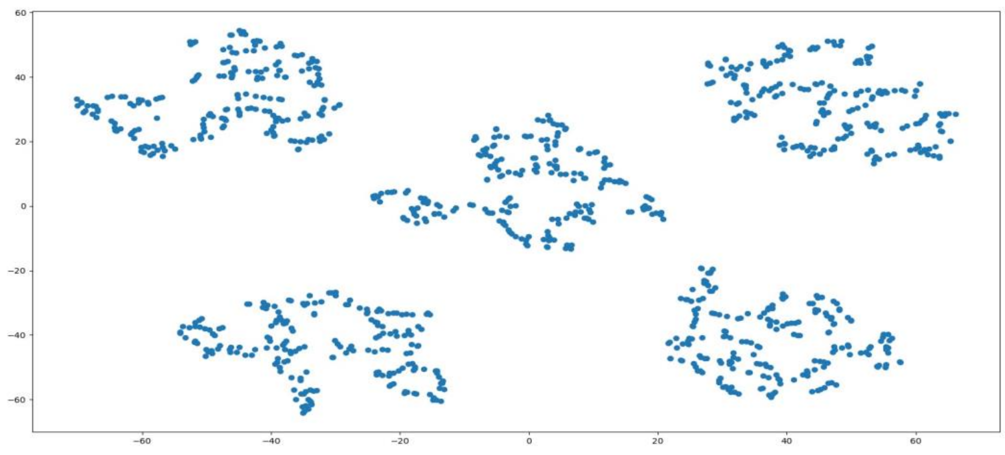
TSNE:

Dataset1)

```
TSNE(n_components=2, perplexity=30, metric="euclidean")
```

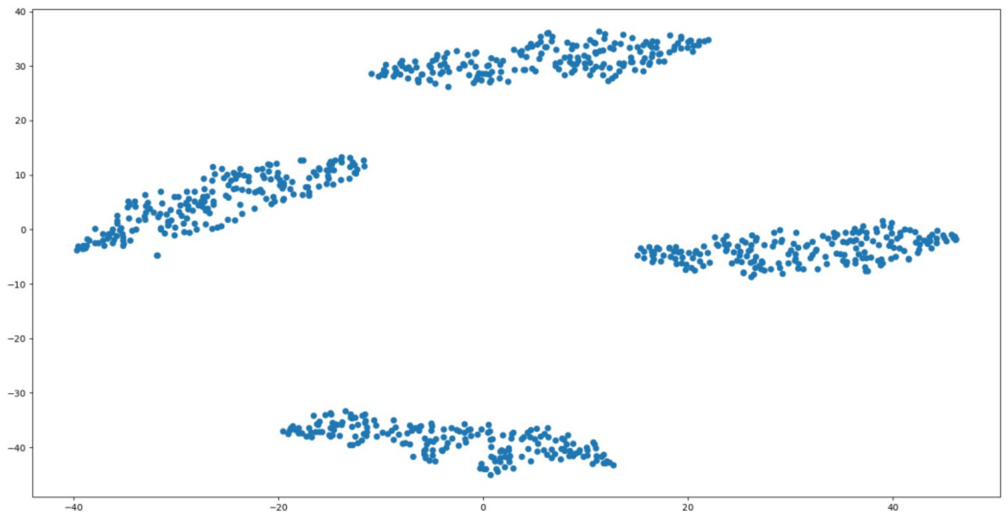


```
TSNE(n_components=2, perplexity=10, metric="cosine")
```

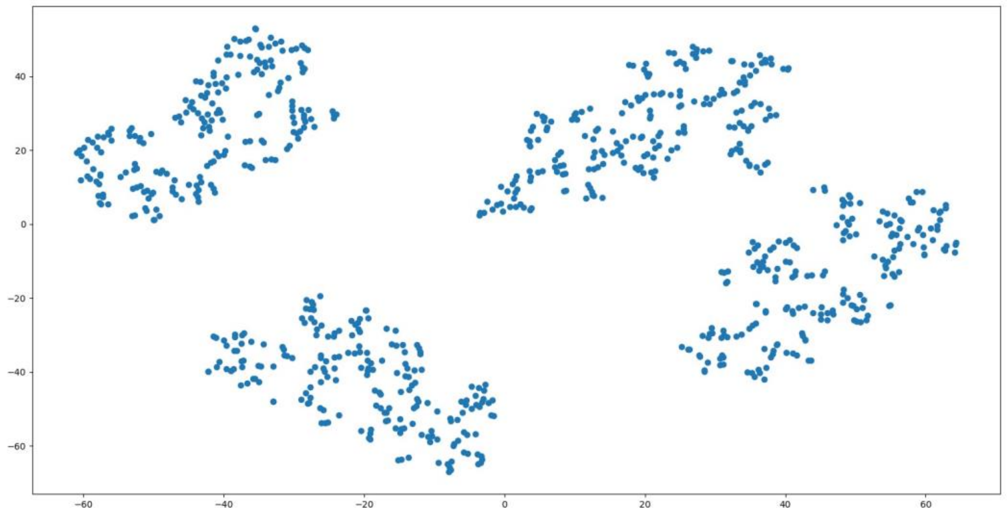


Dataset2)

```
TSNE(n_components=2, perplexity=30, metric="euclidean")
```



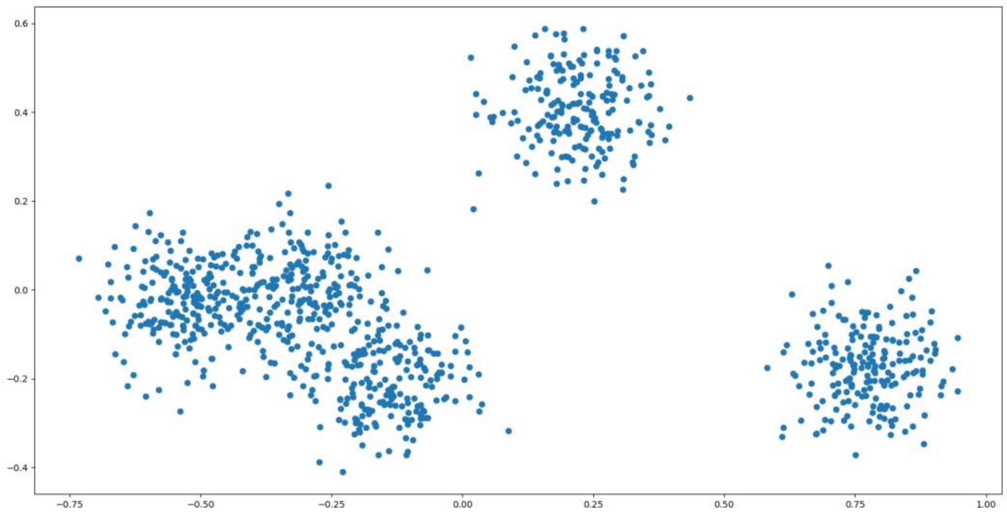
```
TSNE(n_components=2, perplexity=10, metric="cosine")
```



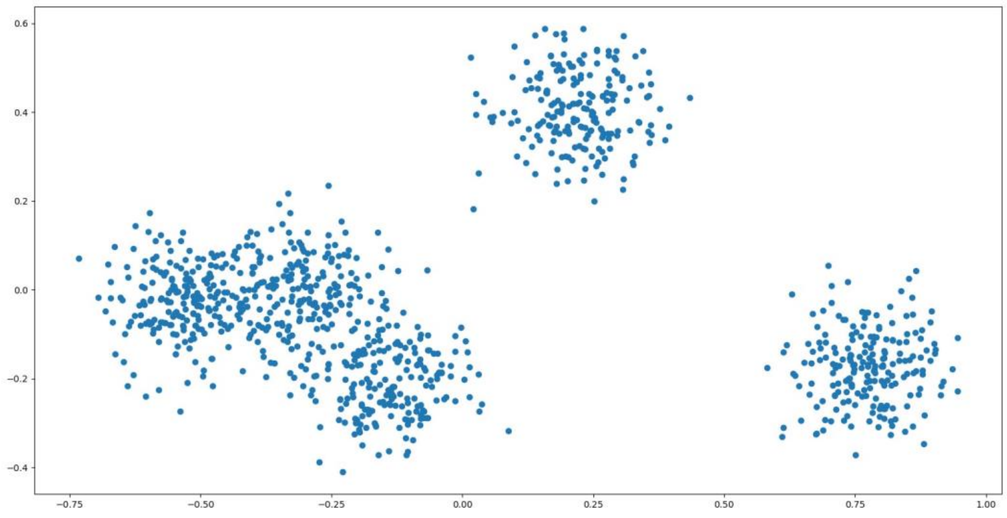
PCA:

Dataset1)

```
method = PCA(n_components=2)
```

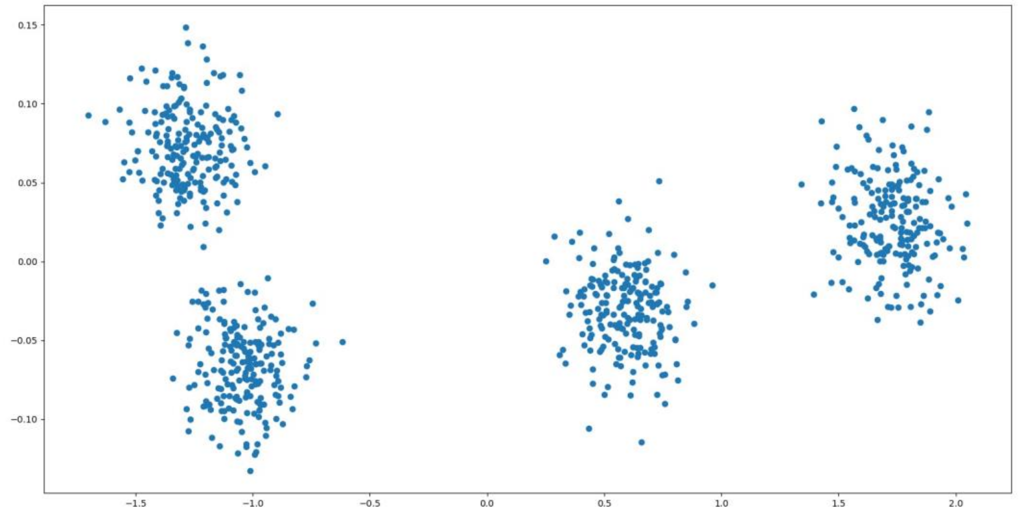


```
PCA(n_components=4)
```

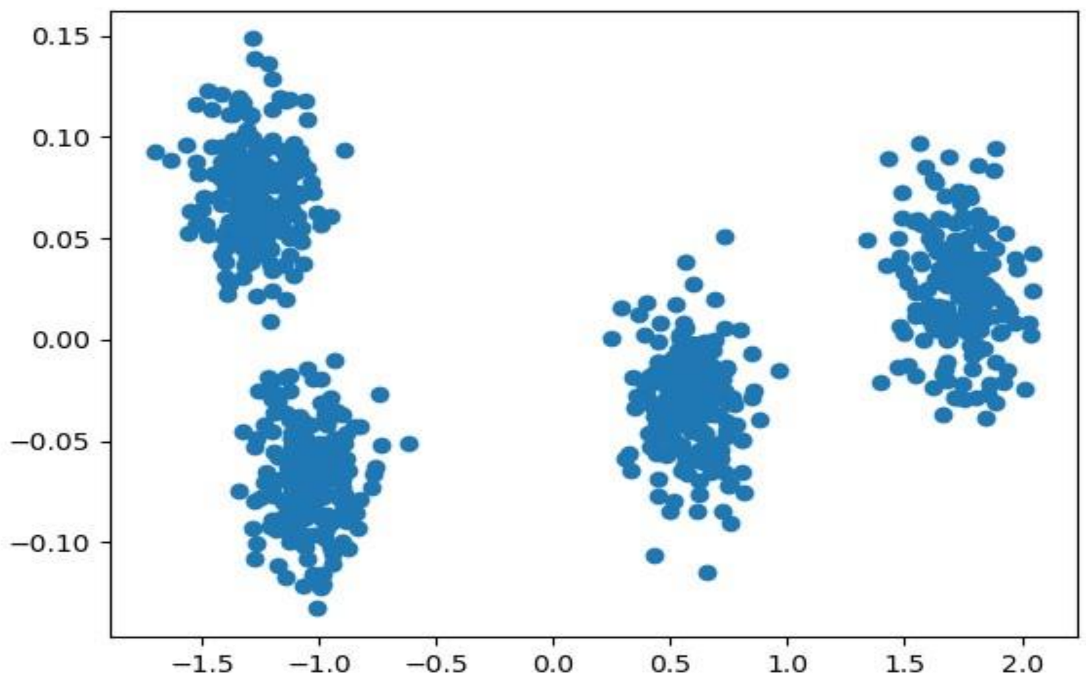


Dataset2)

```
method = PCA(n_components=2)
```



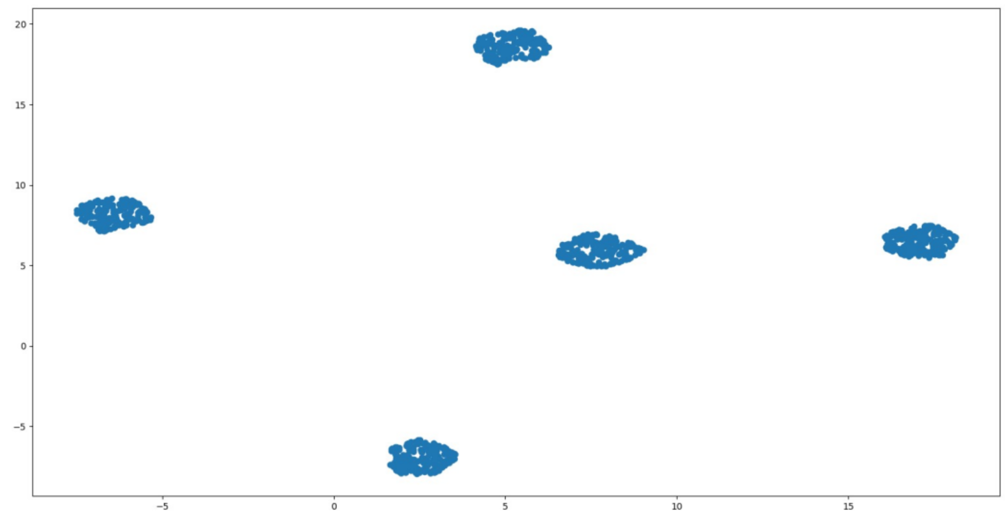
```
PCA(n_components=4)
```



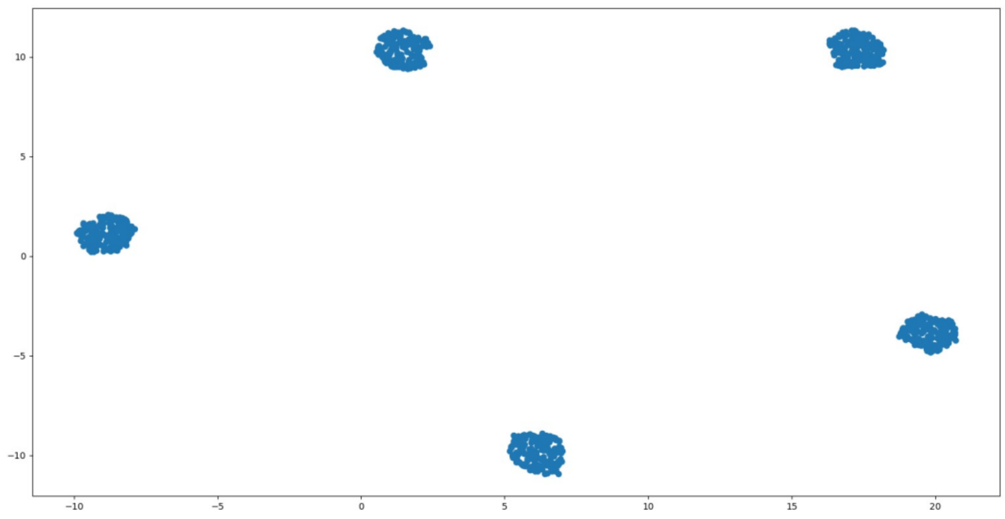
UMAP:

Dataset1)

```
UMAP(n_neighbors=100, n_components=2)
```

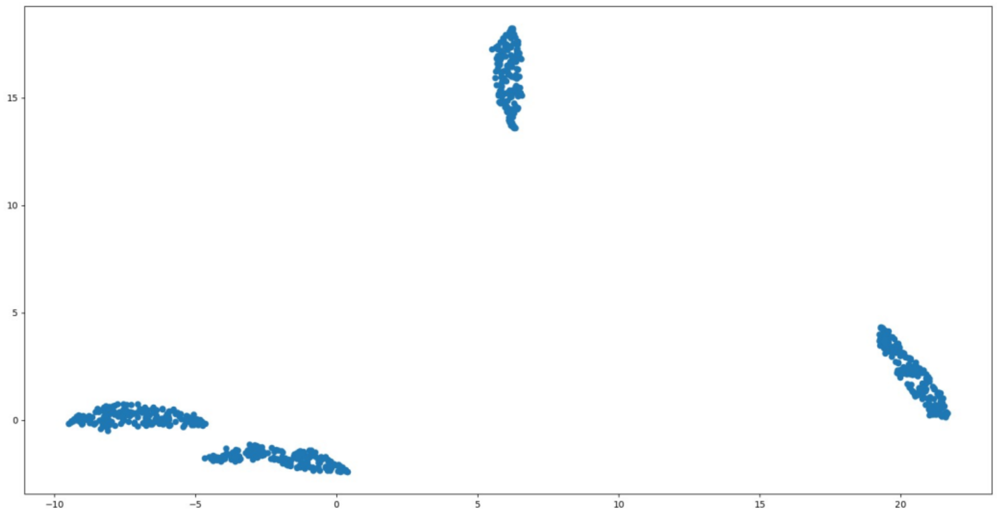


```
UMAP(n_neighbors=50, n_components=2)
```

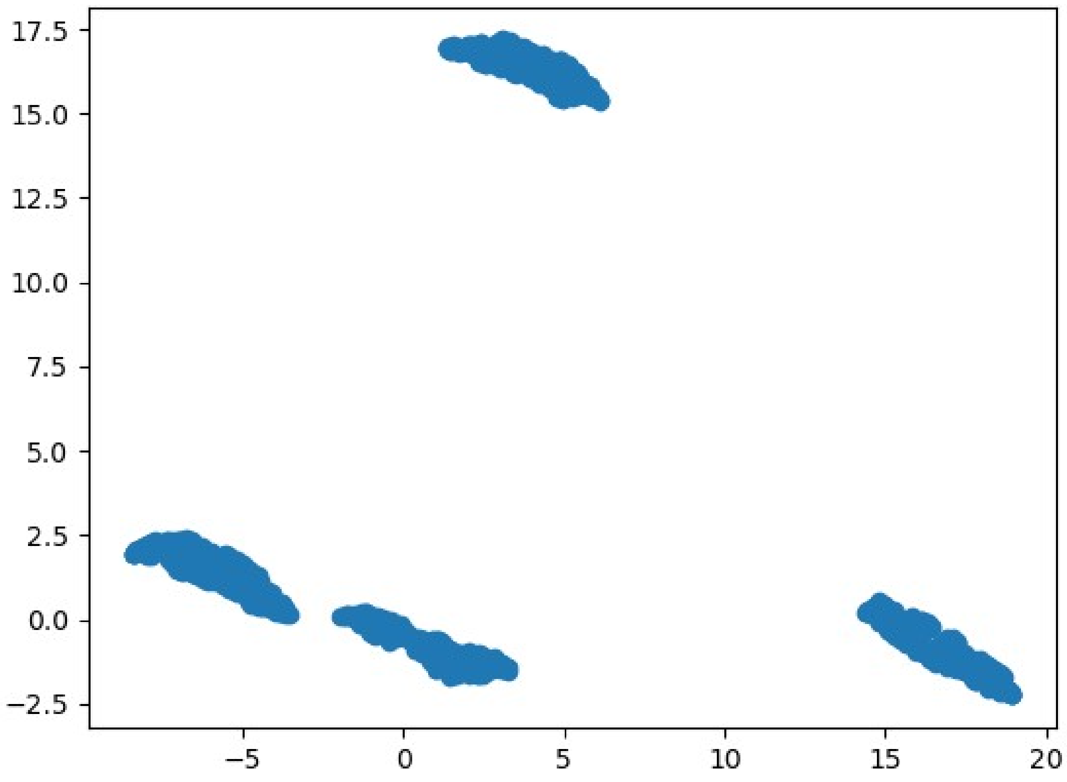


Dataset2)

```
UMAP(n_neighbors=100, n_components=2)
```

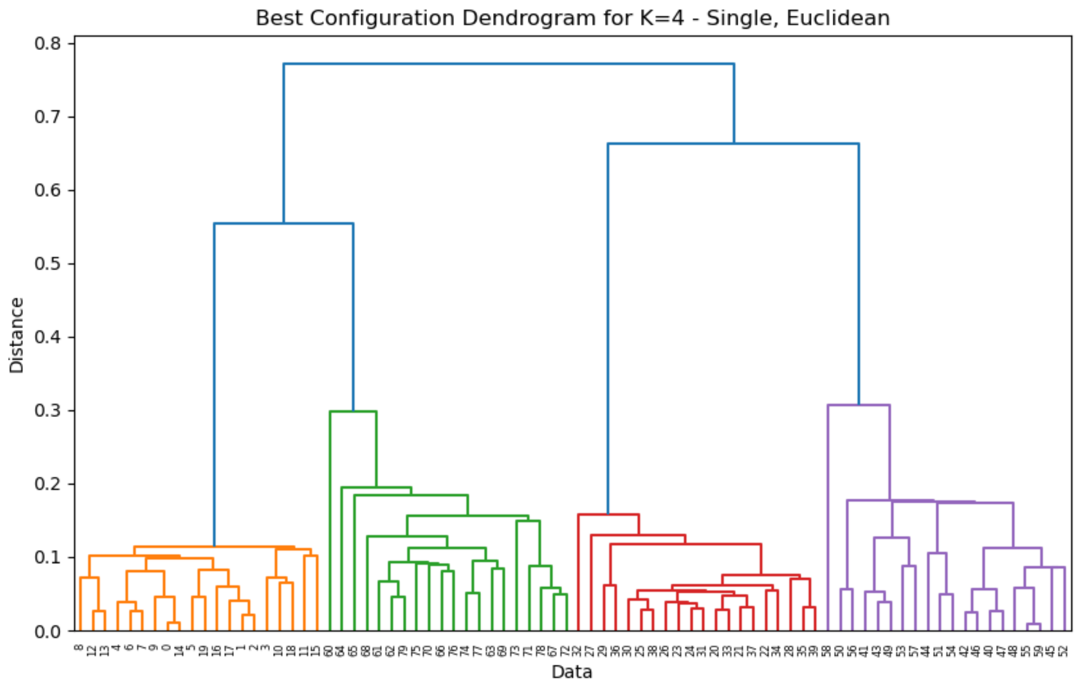


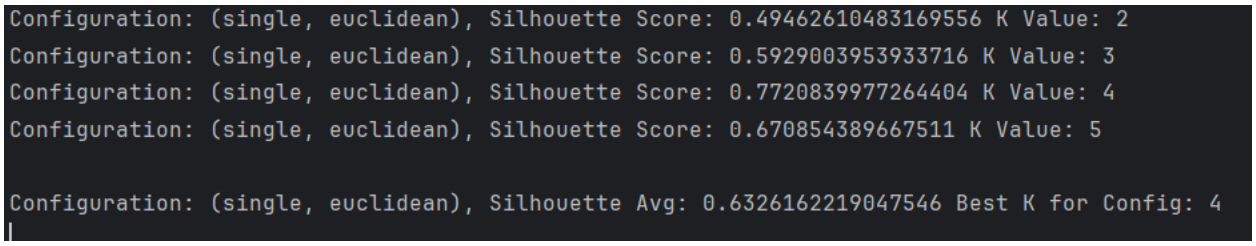
```
UMAP(n_neighbors=50, n_components=2)
```



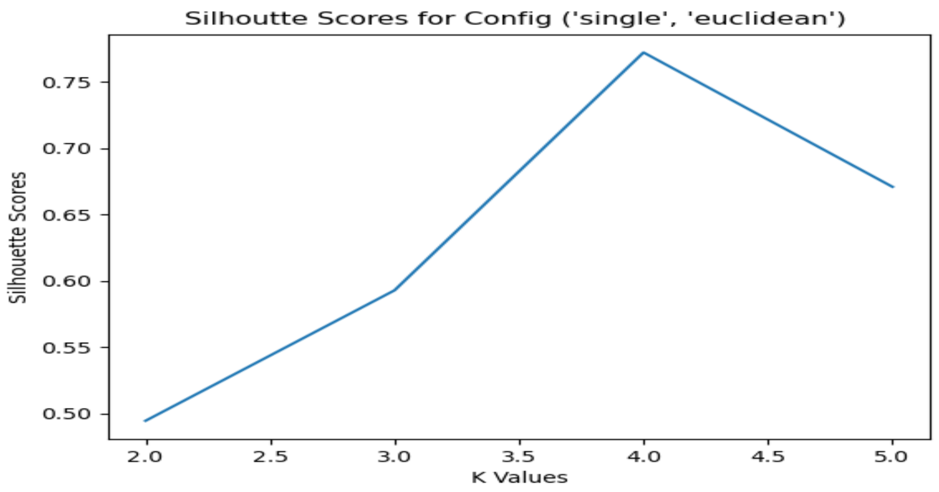
PART3)

CONFIGURATION = (‘SINGLE’, ‘EUCLIDIAN’)



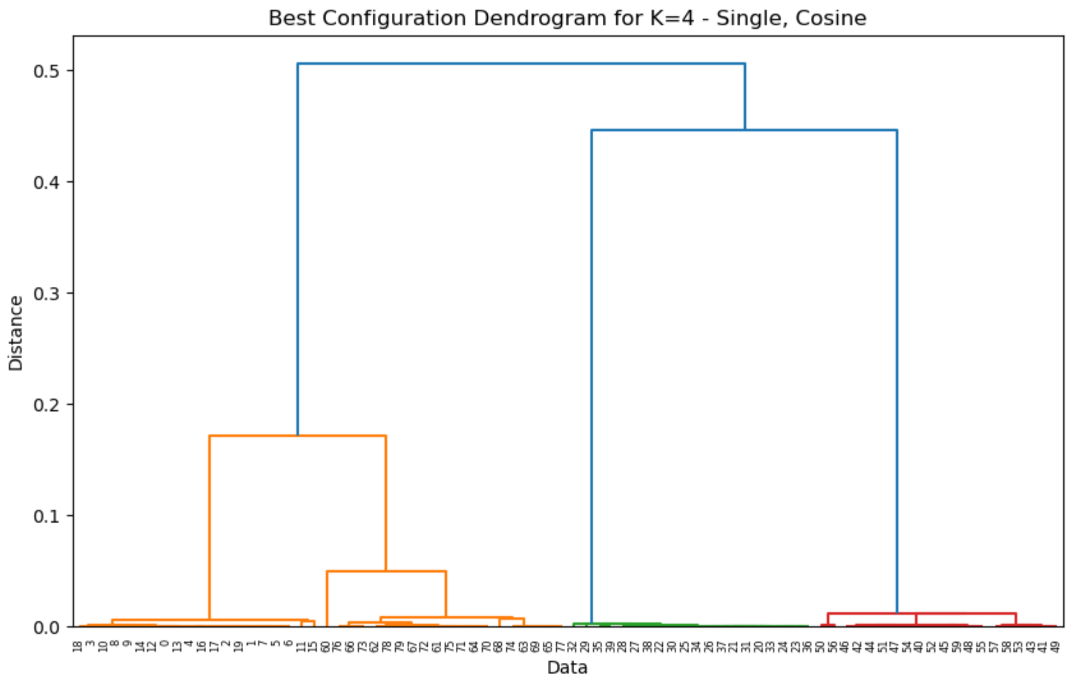


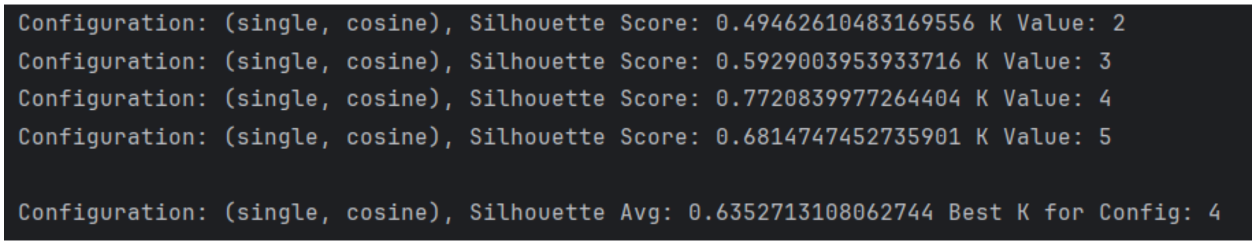
```
Configuration: (single, euclidean), Silhouette Score: 0.49462610483169556 K Value: 2
Configuration: (single, euclidean), Silhouette Score: 0.5929003953933716 K Value: 3
Configuration: (single, euclidean), Silhouette Score: 0.7720839977264404 K Value: 4
Configuration: (single, euclidean), Silhouette Score: 0.670854389667511 K Value: 5

Configuration: (single, euclidean), Silhouette Avg: 0.6326162219047546 Best K for Config: 4
```

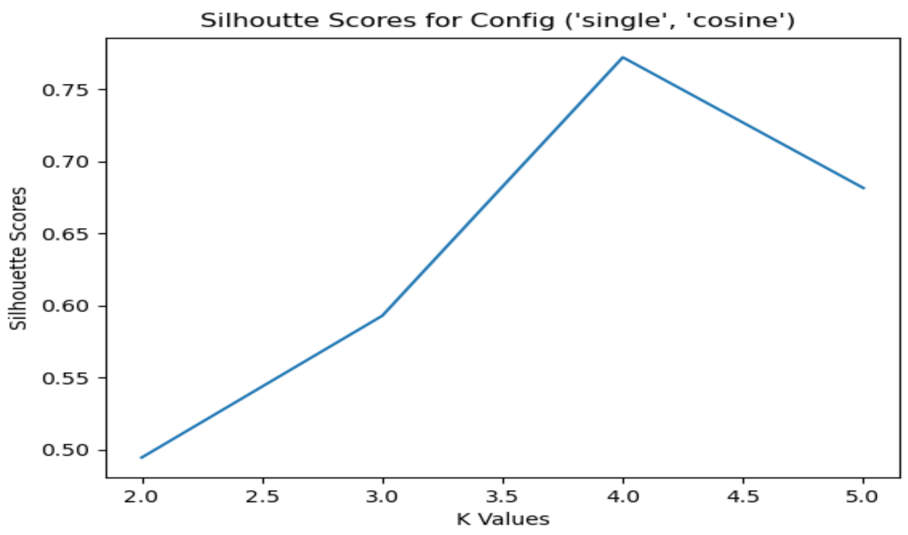


The best K value in the Silhouette Analysis is K=4 for the Config=(‘single’, ‘euclidian’)

CONFIGURATION = ('SINGLE','COSINE')

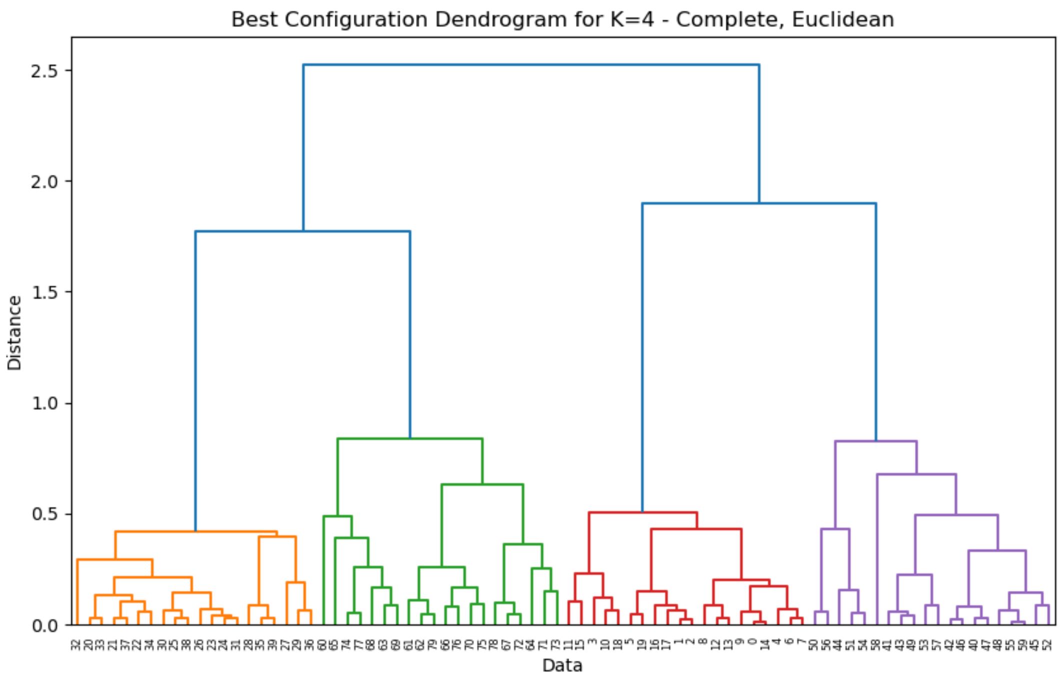


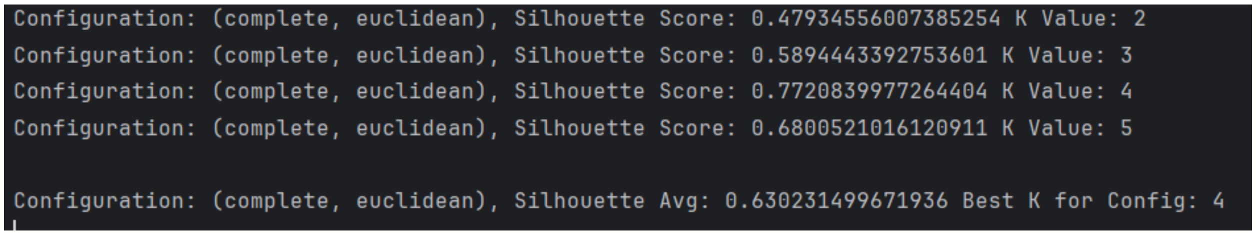

```
Configuration: (single, cosine), Silhouette Score: 0.49462610483169556 K Value: 2
Configuration: (single, cosine), Silhouette Score: 0.5929003953933716 K Value: 3
Configuration: (single, cosine), Silhouette Score: 0.7720839977264404 K Value: 4
Configuration: (single, cosine), Silhouette Score: 0.6814747452735901 K Value: 5

Configuration: (single, cosine), Silhouette Avg: 0.6352713108062744 Best K for Config: 4
```



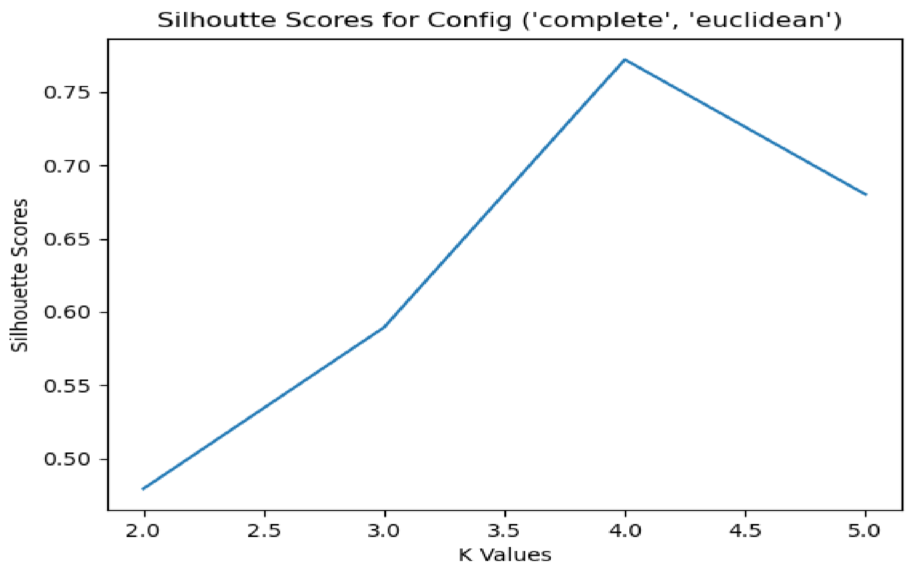
The best K value in the Silhouette Analysis is K=4 for the Config=('single', 'cosine')

CONFIGURATION = (‘COMPLETE’,’EUCLIDIAN’)



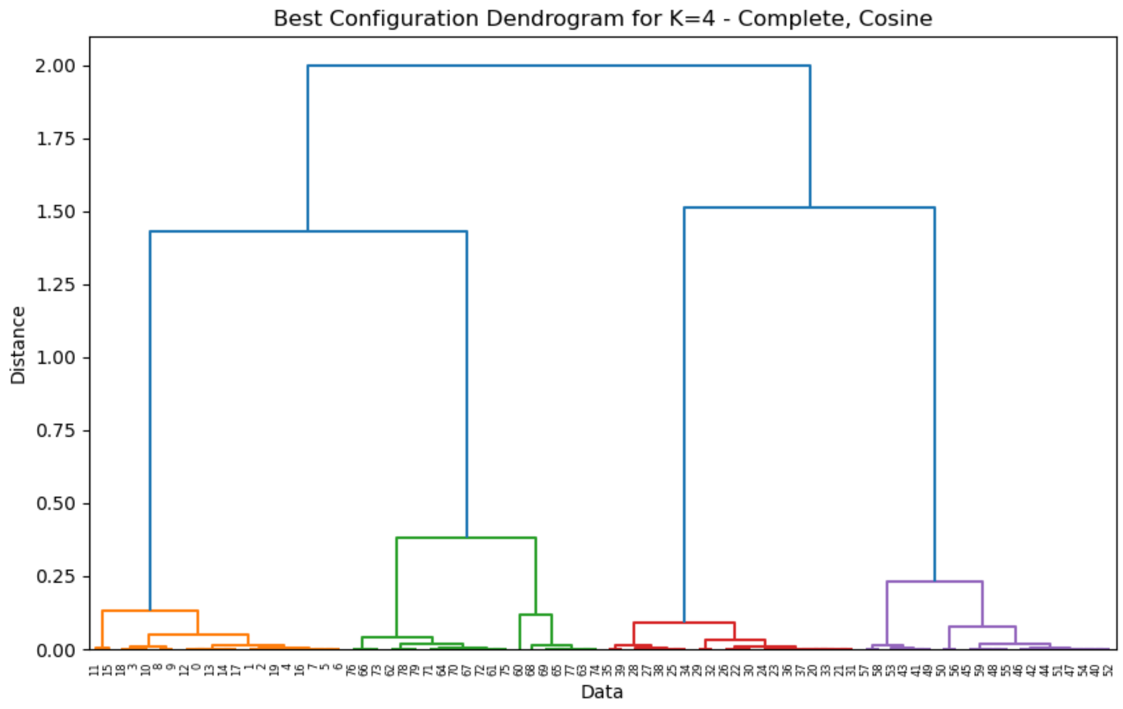


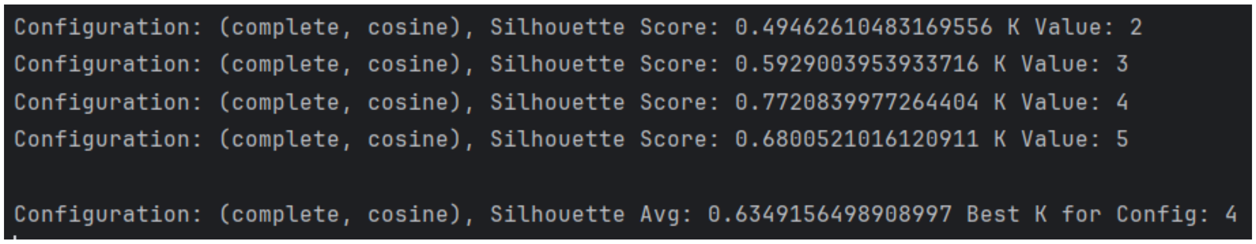
```
Configuration: (complete, euclidean), Silhouette Score: 0.47934556007385254 K Value: 2
Configuration: (complete, euclidean), Silhouette Score: 0.5894443392753601 K Value: 3
Configuration: (complete, euclidean), Silhouette Score: 0.7720839977264404 K Value: 4
Configuration: (complete, euclidean), Silhouette Score: 0.6800521016120911 K Value: 5

Configuration: (complete, euclidean), Silhouette Avg: 0.630231499671936 Best K for Config: 4
```

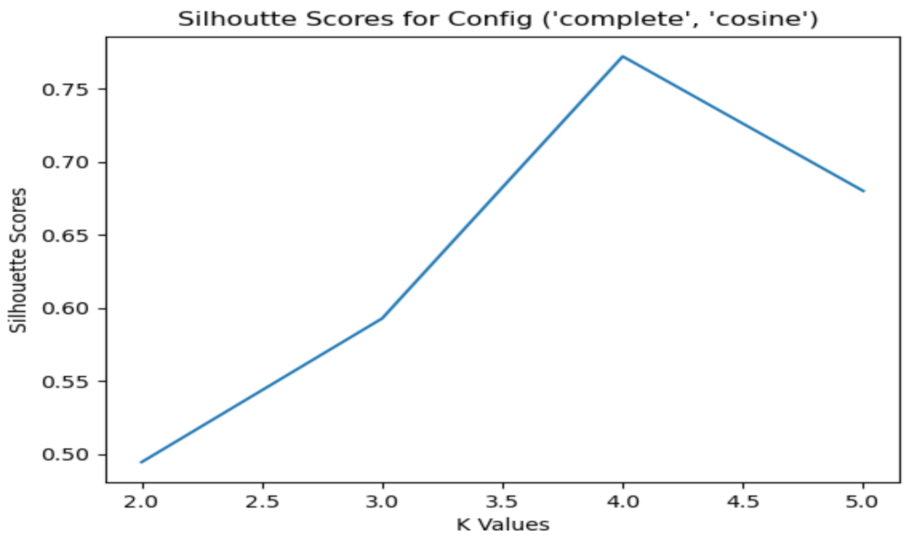


The best K value in the Silhouette Analysis is K=4 for the Config=(‘complete’, ‘euclidean)

CONFIGURATION = (‘COMPLETE’,’COSINE’)

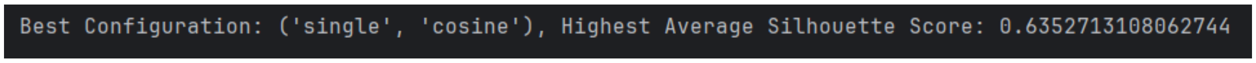



```
Configuration: (complete, cosine), Silhouette Score: 0.49462610483169556 K Value: 2
Configuration: (complete, cosine), Silhouette Score: 0.5929003953933716 K Value: 3
Configuration: (complete, cosine), Silhouette Score: 0.7720839977264404 K Value: 4
Configuration: (complete, cosine), Silhouette Score: 0.6800521016120911 K Value: 5

Configuration: (complete, cosine), Silhouette Avg: 0.6349156498908997 Best K for Config: 4
```



The best K value in the Silhouette Analysis is K=4 for the Config=(‘complete’, ‘cosine)

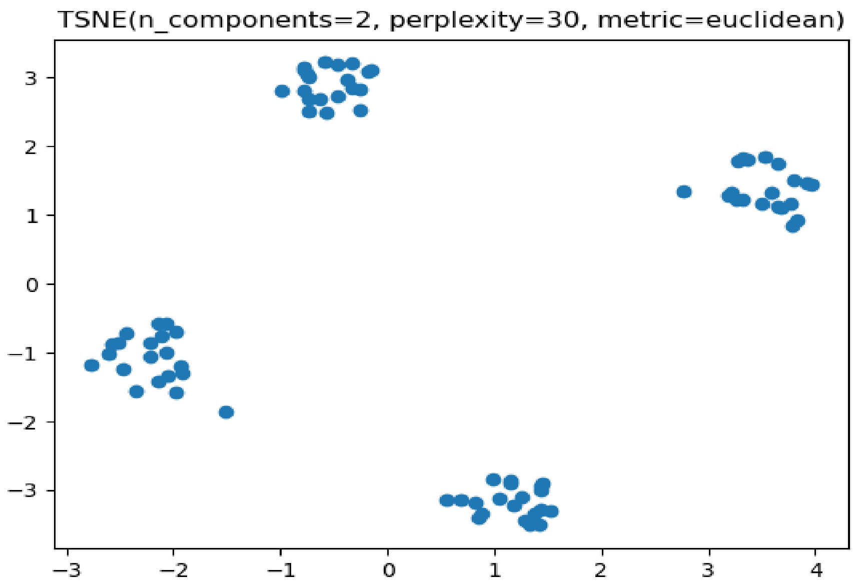
The best configuration is :

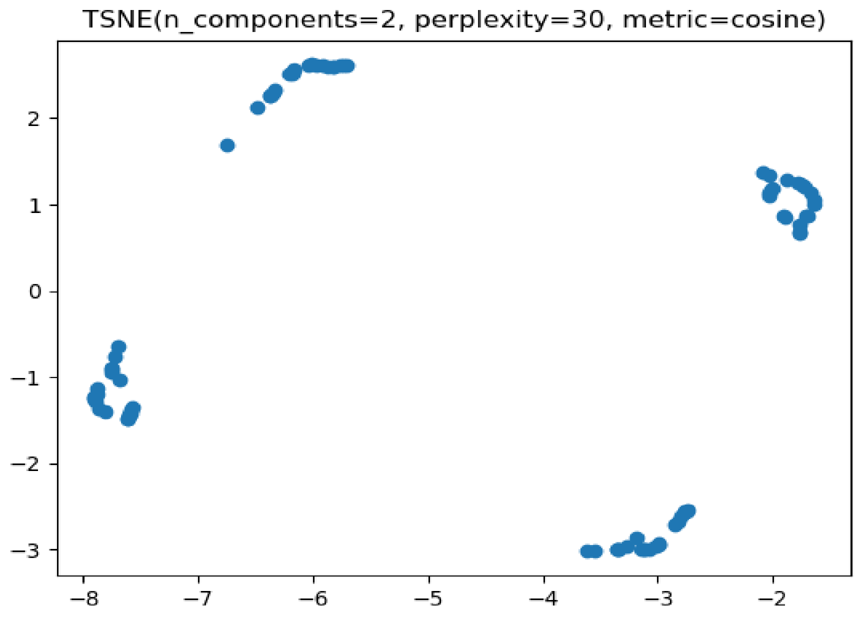


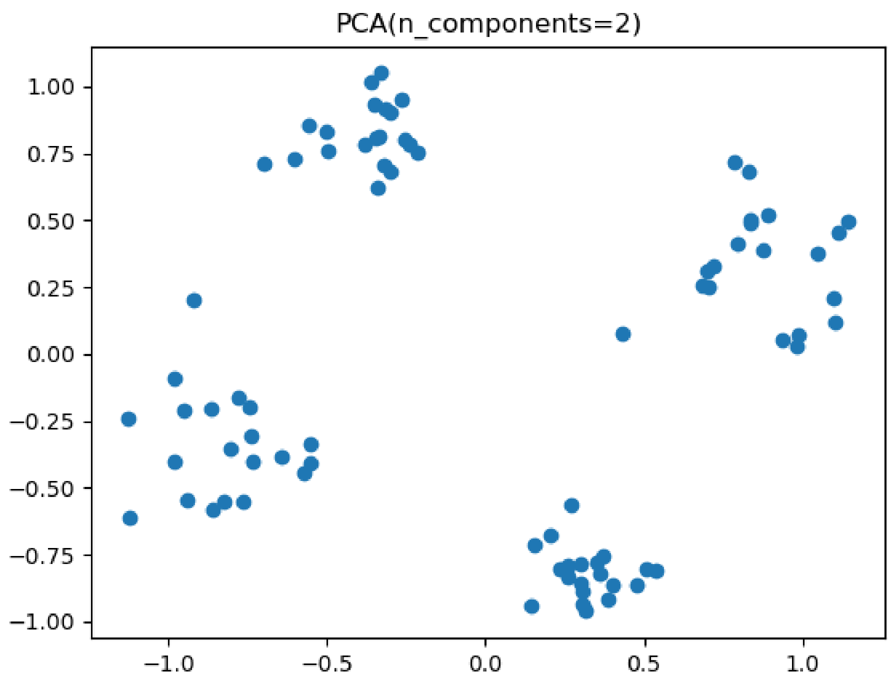
```
Best Configuration: ('single', 'cosine'), Highest Average Silhouette Score: 0.6352713108062744
```

And best K value is 4 which is derived from the Silhouette Analysis.

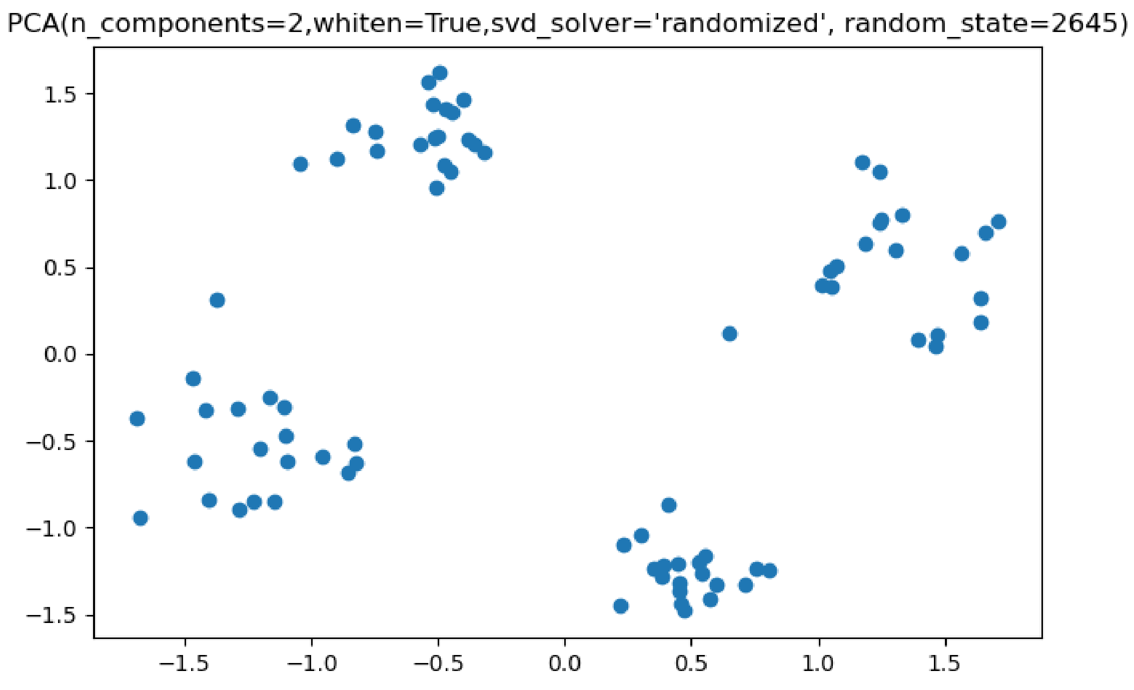
Dimensionality Reduction:

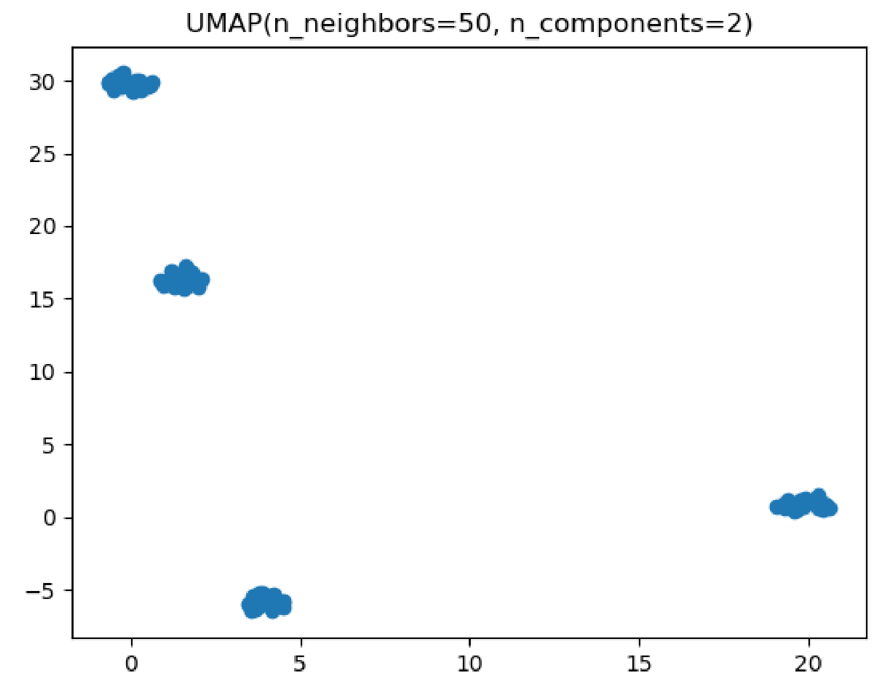


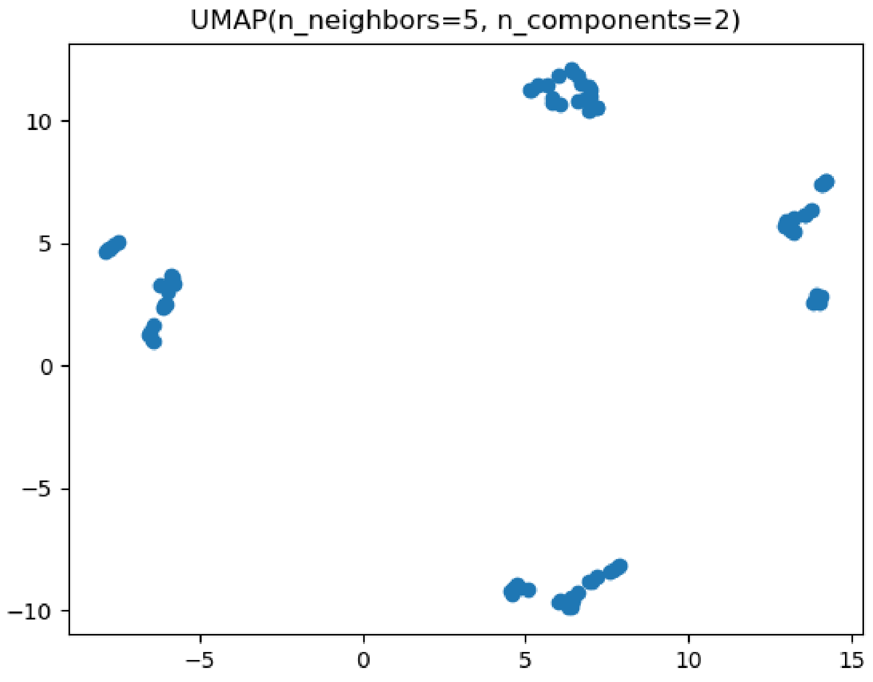




All the other configurations I tried yield the same plot for the PCA method.







The cluster are strict and not scattered around, therefore I found that the :

```
UMAP(n_neighbors=50, n_components=2)
```

Configuration is the best dimensionality reduction that can be used in this analysis.Also, dimensionality reduction results are consistent with the Silhouette Analysis results.