In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
import os
import cv2
import matplotlib.pyplot as plt
from PIL import Image
from skimage import io
from google.colab.patches import cv2_imshow
import numpy as np
import pandas as pd
import math
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from numpy import array
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, classification_report
```

In [3]:
```
DF=pd.read_csv('/content/drive/MyDrive/DM P1/Weed-4class-36/Weed-4class-36-labels.csv'
DF
```

Out[3]:

	Filename	Label	Species
0	20161207-112417-0.jpg	8	Negative
1	20161207-112431-0.jpg	8	Negative
2	20161207-112802-0.jpg	8	Negative
3	20161207-112812-0.jpg	8	Negative
4	20170128-101909-0.jpg	8	Negative
...	...	...	...
13277	20171025-172145-3.jpg	3	Parthenium
13278	20171025-172200-3.jpg	3	Parthenium
13279	20171025-172226-3.jpg	3	Parthenium
13280	20171025-172236-3.jpg	3	Parthenium
13281	20171025-172247-3.jpg	3	Parthenium

13282 rows × 3 columns

In [4]:
```
DF_part=DF.loc[DF['Species']=='Parthenium']
DF_lant=DF.loc[DF['Species']=='Lantana']
DF_snake=DF.loc[DF['Species']=='Snake weed']
DF_siam=DF.loc[DF['Species']=='Siam weed']
```

```
part_imgs=list(DF_part['Filename'])
lant_imgs=list(DF_lant['Filename'])
snake_imgs=list(DF_snake['Filename'])
siam_imgs=list(DF_siam['Filename'])
```

1. Convert the images from the four weed classes (ignoring the negative class images) to grayscale pixel intensity histograms. (You should have done this in Assignment 2) and normalize the histogram dataset.

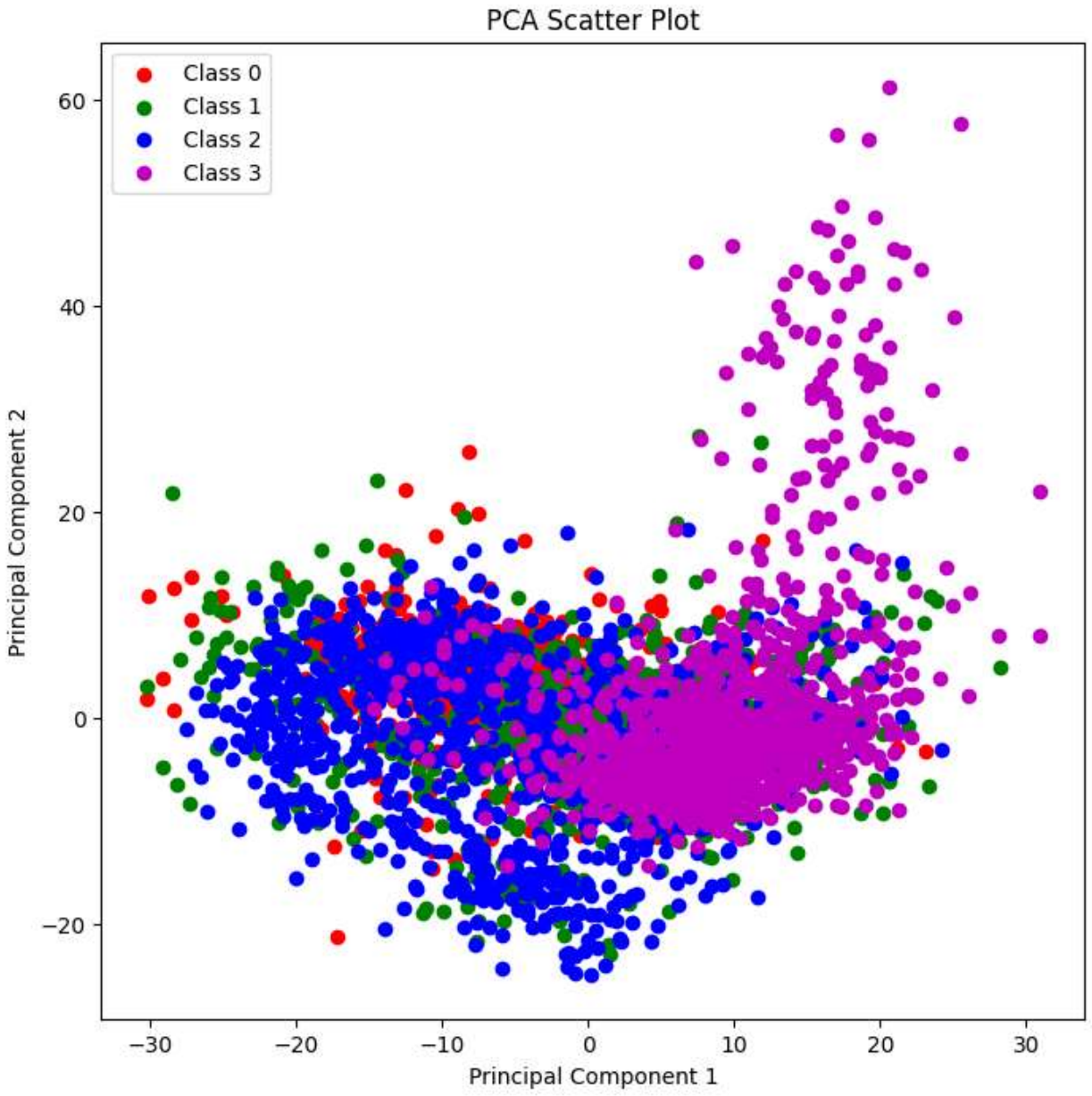
In [5]:
```
part_hist=[]
lant_hist=[]
snake_hist=[]
siam_hist=[]

for img in part_imgs:
  img_gray = cv2.imread('/content/drive/MyDrive/DM P1/Weed-4class-36/'+img,0)
  hist, bins = np.histogram(img_gray.ravel(), 256, [0, 256], density = True)
  part_hist.append(hist)

for img in lant_imgs:
  img_gray = cv2.imread('/content/drive/MyDrive/DM P1/Weed-4class-36/'+img,0)
  hist, bins = np.histogram(img_gray.ravel(), 256, [0, 256], density = True)
  lant_hist.append(hist)

for img in snake_imgs:
  img_gray = cv2.imread('/content/drive/MyDrive/DM P1/Weed-4class-36/'+img,0)
  hist, bins = np.histogram(img_gray.ravel(), 256, [0, 256], density = True)
  snake_hist.append(hist)

for img in siam_imgs:
  img_gray = cv2.imread('/content/drive/MyDrive/DM P1/Weed-4class-36/'+img,0)
  hist, bins = np.histogram(img_gray.ravel(), 256, [0, 256], density = True)
  siam_hist.append(hist)
```

https://www.delftstack.com/howto/matplotlib/normalized-histogram-python/

1. Perform dimension reduction on your histogram dataset to reduce the dimension to 2 (similar to Assignment 1 Question 2(e)).

In [6]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

X = np.concatenate((part_hist, lant_hist, snake_hist, siam_hist), axis=0)

n_samples = len(part_hist) + len(lant_hist) + len(snake_hist) + len(siam_hist)
y = np.zeros(n_samples, dtype=int)

y[len(part_hist):2*len(part_hist)] = 1
y[2*len(part_hist):3*len(part_hist)] = 2
y[3*len(part_hist):] = 3

sc = StandardScaler()
X_scaled = sc.fit_transform(X)
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 8))
colors = ['r', 'g', 'b', 'm']
for i in range(4):
    plt.scatter(X_pca[y == i, 0], X_pca[y == i, 1], color=colors[i], label=f'Class {i}
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Scatter Plot')
plt.legend()
plt.show()
```



https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b

1. Perform clustering using the following approaches on the 2D dataset you preprocessed in Item 2:

• K-mean clustering and its variants for $K = 4$:

(a) K-means clustering: (Use KMeans with init = 'Random') (0.5 point)

(b) KMeans with init='k-means++' (0.5 point)

(c) Bisecting K-means (sklearn.cluster.BisectingKMeans with init = 'Random') (0.5 point)

(d) spectral clustering (sklearn.cluster.SpectralClustering with default parameters) (0.5 point)

In [7]:
```
from sklearn.cluster import KMeans
import numpy as np

kmeans_rand = KMeans(n_clusters=4,  init='random').fit(X_pca)
kmeans_rand.labels_
kmeans_rand.cluster_centers_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[7]:
```
array([[ 16.70697333,  31.19501695],
       [ -0.23695677,  -5.75226239],
       [-12.2688915 ,   4.0187738 ],
       [ 10.30963813,  -0.74044155]])
```

In [8]:
```
from sklearn.cluster import KMeans
import numpy as np

kmeans_plus = KMeans(n_clusters=4,  init='k-means++').fit(X_pca)
kmeans_plus.labels_
kmeans_plus.cluster_centers_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[8]:
```
array([[ 16.70697333,  31.19501695],
       [-12.27840838,   4.01641107],
       [ -0.25690526,  -5.78022131],
       [ 10.26907335,  -0.73299563]])
```

In [9]:
```
from sklearn.cluster import BisectingKMeans
import numpy as np

bisect_means = BisectingKMeans(n_clusters=4, init='random').fit(X_pca)
bisect_means.labels_
bisect_means.cluster_centers_
```

Out[9]:
```
array([[ -6.30972837,  -1.65374541],
       [-15.04476429,   5.13930995],
       [  6.85719675,  -2.89283188],
       [ 17.09757675,  28.18809762]])
```

In [10]:
```
from sklearn.cluster import SpectralClustering
import numpy as np

clustering_spectral = SpectralClustering(n_clusters=4, assign_labels='discretize', ran
```

```
clustering_spectral.labels_
clustering_spectral
```

Out[10]:

SpectralClustering

```
SpectralClustering(assign_labels='discretize', n_clusters=4, random_state=0)
```

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html https://scikit-learn.org/stable/modules/generated/sklearn.cluster.BisectingKMeans.html https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

DBSCAN (0.5 point) – What are the eps and min samples parameter values you used to get 4 clusters? (0.5 point)

In [25]:

```
from sklearn.cluster import DBSCAN
import numpy as np

clustering_dbscan = DBSCAN(eps=3, min_samples=9).fit(X_pca)

clusters=np.unique(clustering_dbscan.labels_)
print(clusters)
clustering_dbscan
```

```
[-1  0  1  2]
```

Out[25]:

DBSCAN

```
DBSCAN(eps=3, min_samples=9)
```

Used eps=3 and min_samples =9

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN.fit

Agglomerative clustering (i.e., hierarchical clustering) - use sklearn.cluster.AgglomerativeClustering with number of clusters set to 4

(a) Single link (MIN), (0.5 point)

(b) Complete link (MAX), (0.5 point)

(c) Group Average, and (0.5 point)

(d) Ward's method (0.5 point)

In [26]:

```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

clustering_single = AgglomerativeClustering(n_clusters=4,linkage='single').fit(X_pca)
clustering_single
clustering_single.labels_
```

Out[26]:

```
array([1, 1, 1, ..., 1, 1, 1])
```

In [27]:
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

clustering_complete = AgglomerativeClustering(n_clusters=4,linkage='complete').fit(X_
clustering_complete
clustering_complete.labels_
```

Out[27]:
```
array([1, 0, 0, ..., 1, 1, 1])
```

In [28]:
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

clustering_average = AgglomerativeClustering(n_clusters=4,linkage='average').fit(X_pca
clustering_average
clustering_average.labels_
```

Out[28]:
```
array([3, 3, 3, ..., 1, 1, 1])
```

In [29]:
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

clustering_ward = AgglomerativeClustering(n_clusters=4,linkage='ward').fit(X_pca)
clustering_ward
clustering_ward.labels_
```

Out[29]:
```
array([0, 1, 1, ..., 0, 3, 0])
```

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn-cluster-agglomerativeclustering

1. For all the methods in Item 3:

(a) Perform clustering performance evaluation using Fowlkes-Mallows index (sklearn.metrics.fowlkes mallows score). Compute the Fowlkes-Mallows index for each method on the 2D dataset. (0.5 point)

In [34]:
```
from sklearn.metrics.cluster import fowlkes_mallows_score

fowlkes_kmeans_rand=fowlkes_mallows_score(y, kmeans_rand.labels_)
fowlkes_kmeans_plus=fowlkes_mallows_score(y, kmeans_plus.labels_)
fowlkes_bisect_means=fowlkes_mallows_score(y, bisect_means.labels_)
fowlkes_clustering_spectral=fowlkes_mallows_score(y, clustering_spectral.labels_)
fowlkes_clustering_dbscan=fowlkes_mallows_score(y, clustering_dbscan.labels_)
fowlkes_clustering_single=fowlkes_mallows_score(y, clustering_single.labels_)
fowlkes_clustering_complete=fowlkes_mallows_score(y, clustering_complete.labels_)
fowlkes_clustering_average=fowlkes_mallows_score(y, clustering_average.labels_)
fowlkes_clustering_ward=fowlkes_mallows_score(y, clustering_ward.labels_)
```

In [35]:
```
print(fowlkes_kmeans_rand)
print(fowlkes_kmeans_plus)
print(fowlkes_bisect_means)
print(fowlkes_clustering_spectral)
print(fowlkes_clustering_dbscan)
```

```
print(fowlkes_clustering_single)
print(fowlkes_clustering_complete)
print(fowlkes_clustering_average)
print(fowlkes_clustering_ward)
```

```
0.37690811113123185
0.37669393173495486
0.39199675792268396
0.49898629429359864
0.4842665658902417
0.49936857944268553
0.4011306195885118
0.441364015868573
0.3656711871223799
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fowlkes_mallows_score.html

(b) Perform clustering performance evaluation using Silhouette Coefficient (sklearn.metrics.silhouette score). Compute the Silhouette Coefficient for each method. (0.5 point)

In [38]:
```
from sklearn.metrics import silhouette_score

silhouette_kmeans_rand=silhouette_score(X_pca, kmeans_rand.labels_)
silhouette_kmeans_plus=silhouette_score(X_pca, kmeans_plus.labels_)
silhouette_bisect_means=silhouette_score(X_pca, bisect_means.labels_)
silhouette_clustering_spectral=silhouette_score(X_pca, clustering_spectral.labels_)
silhouette_clustering_dbscan=silhouette_score(X_pca, clustering_dbscan.labels_)
silhouette_clustering_single=silhouette_score(X_pca, clustering_single.labels_)
silhouette_clustering_complete=silhouette_score(X_pca, clustering_complete.labels_)
silhouette_clustering_average=silhouette_score(X_pca, clustering_average.labels_)
silhouette_clustering_ward=silhouette_score(X_pca, clustering_ward.labels_)
```

In [39]:
```
print(silhouette_kmeans_rand)
print(silhouette_kmeans_plus)
print(silhouette_bisect_means)
print(silhouette_clustering_spectral)
print(silhouette_clustering_dbscan)
print(silhouette_clustering_single)
print(silhouette_clustering_complete)
print(silhouette_clustering_average)
print(silhouette_clustering_ward)
```

```
0.366036855295145
0.3660297513791778
0.3493864562042678
0.27167625348833463
0.5131723540247091
0.42907642223376913
0.3604195084635566
0.4173219065208064
0.3372147221868336
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

(c) What is the main difference between Fowlkes-Mallows index and Silhouette Coefficient? (0.5 point)

Both the Fowlkes-Mallows (FM) index and the Silhouette Coefficient are used to measure how well a clustering algorithm works, but their methods and calculations are different.

The FM index measures how similar the cluster assignments made by the clustering algorithm are to the real cluster assignments. It is based on the number of pairs of data points that both the true labels and the clustering algorithm put in the same cluster, divided by the total number of pairs of data points. A better performance at clustering is shown by a higher FM index value.

On the other hand, the Silhouette Coefficient is a way to measure how well each data point fits into its cluster. It figures out the average distance between a data point and all the other points in its cluster (a measure of cohesion) and the average distance between a data point and all the other points in the cluster next to it (a measure of separation). The Silhouette Coefficient is then found by taking the difference between cohesion and separation and dividing it by the larger of the two numbers. A higher Silhouette Coefficient value means that the data points are well clustered and there is a clear separation between clusters.

In short, the FM index measures how similar the true labels and clustering results are as a whole, while the Silhouette Coefficient measures how well each data point fits into its own cluster.

(d) Rank the methods from the best to the worst for our dataset based on Fowlkes-Mallows index. (0.5 point)

1. AgglomerativeClustering with single link
2. Spectral clustering
3. DBSCAN clustering
4. AgglomerativeClustering with average link
5. AgglomerativeClustering with complete link
6. BisectingKMeans clustering
7. KMeans clustering with 'random' init
8. KMeans clustering with 'k-means++' init
9. AgglomerativeClustering with ward link

(e) Rank the methods from the best to the worst for our dataset based on Silhouette Coefficient. (0.5 point)

1.DBSCAN clustering

2.Agglomerative Clustering with single link

3.Agglomerative Clustering with average link

4.KMeans Clustering with 'random' init

5.KMeans Clustering with 'k-means++' init

6.Agglomerative Clustering with complete link

7.BisectingKMeans clustering

8.Spectral clustering

9.Agglomerative Clustering with ward link

(f) Which one do you think is better for our dataset? Fowlkes-Mallows index or Silhouette Coefficient? Why? (0.5 point)

Fowlkes-Mallows index is best for our dataset, as it got higher scores for all the cluserings performed. In Fowlkes-Mallows index compared the obtained clustering with a reference clustering. In Silhouette Coefficient can know how well each individual sample is being clustered. So, overall for our dataset, Fowlkes-Mallows index is better

1. Perform K-mean clustering and its variants for K = 4 in Item 3 (i.e., 4 methods need to be performed) on the processed dataset in Item 1 (i.e., perform clustering on the dataset without dimensionality reduction). (0.5 point) (a) Do 4(a) (0.25 point)

In [ ]:
```
from sklearn.cluster import KMeans
import numpy as np

kmeans_rand_x = KMeans(n_clusters=4,  init='random').fit(X)
kmeans_rand_x.labels_
kmeans_rand_x.cluster_centers_
```

In [41]:
```
from sklearn.cluster import KMeans
import numpy as np

kmeans_plus_x = KMeans(n_clusters=4,  init='k-means++').fit(X)
kmeans_plus_x.labels_
kmeans_plus_x.cluster_centers_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[41]:
```
array([[0.0005906 , 0.00012972, 0.00015423, ..., 0.00079449, 0.00077607,
        0.00706534],
       [0.0032926 , 0.00079825, 0.00097182, ..., 0.00168141, 0.00167373,
        0.01648071],
       [0.0038757 , 0.0009483 , 0.00115098, ..., 0.00277738, 0.00278914,
        0.02864298],
       [0.00066177, 0.00017803, 0.00021885, ..., 0.00126839, 0.00124599,
        0.01217128]])
```

In [42]:
```
from sklearn.cluster import BisectingKMeans
import numpy as np

bisect_means_x = BisectingKMeans(n_clusters=4, init='random').fit(X)
bisect_means_x.labels_
bisect_means_x.cluster_centers_
```

Out[42]:
```
array([[0.00056053, 0.00012278, 0.00014534, ..., 0.00079001, 0.00077209,
        0.00700672],
       [0.00070292, 0.00018659, 0.0002303 , ..., 0.00121838, 0.00118999,
        0.01160227],
       [0.00330955, 0.00080564, 0.00097831, ..., 0.00193374, 0.00193132,
        0.01931105],
       [0.00306187, 0.00076604, 0.0009417 , ..., 0.00308139, 0.00311776,
        0.03207983]])
```

In [43]:
```
from sklearn.cluster import SpectralClustering
import numpy as np

clustering_spectral_x = SpectralClustering(n_clusters=4, assign_labels='discretize', r
clustering_spectral_x.labels_
clustering_spectral_x
```

Out[43]:

SpectralClustering

```
SpectralClustering(assign_labels='discretize', n_clusters=4, random_state=0)
```

(a) Do 4(a) (0.25 point)

In [44]:
```
from sklearn.metrics.cluster import fowlkes_mallows_score

fowlkes_kmeans_rand_x=fowlkes_mallows_score(y, kmeans_rand_x.labels_)
fowlkes_kmeans_plus_x=fowlkes_mallows_score(y, kmeans_plus_x.labels_)
fowlkes_bisect_means_x=fowlkes_mallows_score(y, bisect_means_x.labels_)
fowlkes_clustering_spectral_x=fowlkes_mallows_score(y, clustering_spectral_x.labels_)
```

In [48]:
```
print(fowlkes_kmeans_rand_x)
print(fowlkes_kmeans_plus_x)
print(fowlkes_bisect_means_x)
print(fowlkes_clustering_spectral_x)
```

```
0.3745918605040608
0.3748006910443745
0.3947213419755175
0.3854545778594182
```

(b) Do 4(b) (0.25 point)

In [49]:
```
from sklearn.metrics import silhouette_score

silhouette_kmeans_rand_x=silhouette_score(X, kmeans_rand_x.labels_)
silhouette_kmeans_plus_x=silhouette_score(X, kmeans_plus_x.labels_)
silhouette_bisect_means_x=silhouette_score(X, bisect_means_x.labels_)
silhouette_clustering_spectral_x=silhouette_score(X, clustering_spectral_x.labels_)
```

In [50]:
```
print(silhouette_kmeans_rand_x)
print(silhouette_kmeans_plus_x)
print(silhouette_bisect_means_x)
print(silhouette_clustering_spectral_x)
```

```
0.22647877053442128
0.22670854440598096
0.23068004938616424
0.19748974935535324
```

(c) Do 4(d) (0.25 point)

1. BisectingKMeans clustering
2. Spectral clustering
3. KMeans clustering with 'k-means++' init
4. KMeans clustering with 'random' init

(d) Do 4(e) (0.25 point)

1. BisectingKMeans clustering
2. KMeans clustering with 'k-means++' init
3. KMeans clustering with 'random' init
4. Spectral clustering

(e) Are the methods performing better with the original dataset or the one with dimension reduced? Explain your observation. (0.5 point)

As, dimensionality reduction can improve clustering performance by reducing noise and redundancy. Dimension reduced dataset performed better compared to the original dataset for all the methods. As, the score for dimension reduced dataset is higher compared to the original dataset.