Partioning Around Mediods on Iris Dataset on UCI Repository

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
```

Reading Datasets

In [2]:
```
data = pd.read_csv('iris.txt', delimiter=',', names = ['sepal length(cm)','sepal wid
data.head()
```

Out[2]:

	sepal length(cm)	sepal width(cm)	petal_length(cm)	petal_width(cm)	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Data Manipulation

In [3]:
```
data['target'] = data['target'].replace({'Iris-setosa': 0, 'Iris-versicolor': 1, 'Ir
data.head()
```

Out[3]:

	sepal length(cm)	sepal width(cm)	petal_length(cm)	petal_width(cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [4]:
```
x_data = data.iloc[:, :4]
print(x_data)
```

```
     sepal length(cm)  sepal width(cm)  petal_length(cm)  petal_width(cm)
```

```
0            5.1          3.5          1.4          0.2
1            4.9          3.0          1.4          0.2
2            4.7          3.2          1.3          0.2
3            4.6          3.1          1.5          0.2
4            5.0          3.6          1.4          0.2
..           ...          ...          ...          ...
145          6.7          3.0          5.2          2.3
146          6.3          2.5          5.0          1.9
147          6.5          3.0          5.2          2.0
148          6.2          3.4          5.4          2.3
149          5.9          3.0          5.1          1.8

[150 rows x 4 columns]
```

In [5]:

```
y_data = data.iloc[:, 4]
print(y_data)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int64
```

In [6]:

```
x_data.isnull().sum()
```

Out[6]:

```
sepal length(cm)    0
sepal width(cm)     0
petal_length(cm)    0
petal_width(cm)     0
dtype: int64
```

Data Normalization

In [7]:

```
scaler = StandardScaler()
```

In [8]:

```
x_transformed = pd.DataFrame(scaler.fit_transform(x_data), columns=x_data.columns)
x_transformed.head()
```

Out[8]:

	sepal length(cm)	sepal width(cm)	petal_length(cm)	petal_width(cm)
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

Dimensionality Reduction

In [9]:
```
pca = PCA(n_components=3)
principalComponents = pca.fit_transform(x_transformed)
df = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'p
df.head()
```

Out[9]:

	principal component 1	principal component 2	principal component 3
0	-2.264542	0.505704	-0.121943
1	-2.086426	-0.655405	-0.227251
2	-2.367950	-0.318477	0.051480
3	-2.304197	-0.575368	0.098860
4	-2.388777	0.674767	0.021428

In [10]:
```
datapoints = df.values
m, f = datapoints.shape
k = 3
print(datapoints)
```

```
[[-2.26454173e+00  5.05703903e-01 -1.21943348e-01]
 [-2.08642550e+00 -6.55404729e-01 -2.27250832e-01]
 [-2.36795045e+00 -3.18477311e-01  5.14796236e-02]
 [-2.30419716e+00 -5.75367713e-01  9.88604444e-02]
 [-2.38877749e+00  6.74767397e-01  2.14278490e-02]
 [-2.07053681e+00  1.51854856e+00  3.06842583e-02]
 [-2.44571134e+00  7.45626750e-02  3.42197636e-01]
 [-2.23384186e+00  2.47613932e-01 -8.25744645e-02]
 [-2.34195768e+00 -1.09514636e+00  1.53562399e-01]
 [-2.18867576e+00 -4.48629048e-01 -2.46559522e-01]
 [-2.16348656e+00  1.07059558e+00 -2.64009373e-01]
 [-2.32737775e+00  1.58587455e-01  1.00165616e-01]
 [-2.22408272e+00 -7.09118158e-01 -2.23214514e-01]
 [-2.63971626e+00 -9.38281982e-01  1.89570030e-01]
 [-2.19229151e+00  1.88997851e+00 -4.69480095e-01]
 [-2.25146521e+00  2.72237108e+00  3.26037967e-02]
 [-2.20275048e+00  1.51375028e+00 -1.36349158e-03]
 [-2.19017916e+00  5.14304308e-01 -3.86155949e-02]
 [-1.89407429e+00  1.43111071e+00 -3.70742834e-01]
 [-2.33994907e+00  1.15803343e+00  1.37417719e-01]
 [-1.91455639e+00  4.30465163e-01 -4.16006875e-01]
 [-2.20464540e+00  9.52457317e-01  1.64738346e-01]
 [-2.77416979e+00  4.89517027e-01  3.38836384e-01]
 [-1.82041156e+00  1.06750793e-01  4.00614724e-02]
 [-2.22821750e+00  1.62186163e-01  1.24201428e-01]
 [-1.95702401e+00 -6.07892567e-01 -2.98591029e-01]
 [-2.05206331e+00  2.66014312e-01  9.20929788e-02]
 [-2.16819365e+00  5.52016495e-01 -2.01295482e-01]
 [-2.14030596e+00  3.36640409e-01 -2.65314545e-01]
 [-2.26879019e+00 -3.14878603e-01  7.55154360e-02]
 [-2.14455443e+00 -4.83942097e-01 -6.78557607e-02]
 [-1.83193810e+00  4.45266836e-01 -2.65375244e-01]
 [-2.60820287e+00  1.82847519e+00  5.14195182e-02]
 [-2.43795086e+00  2.18539162e+00 -7.93497549e-02]
 [-2.18867576e+00 -4.48629048e-01 -2.46559522e-01]
 [-2.21111990e+00 -1.84337811e-01 -2.18624528e-01]
 [-2.04441652e+00  6.84956426e-01 -4.79411570e-01]
 [-2.18867576e+00 -4.48629048e-01 -2.46559522e-01]
 [-2.43595220e+00 -8.82169415e-01  2.01557587e-01]
 [-2.17054720e+00  2.92726955e-01 -1.69938536e-01]
 [-2.28652724e+00  4.67991716e-01  4.07365390e-02]
 [-1.87170722e+00 -2.32769161e+00 -1.94528610e-01]
 [-2.55783442e+00 -4.53816380e-01  3.13571838e-01]
 [-1.96427929e+00  4.97391640e-01  3.14755610e-01]
 [-2.13337283e+00  1.17143211e+00  2.52793222e-01]
```

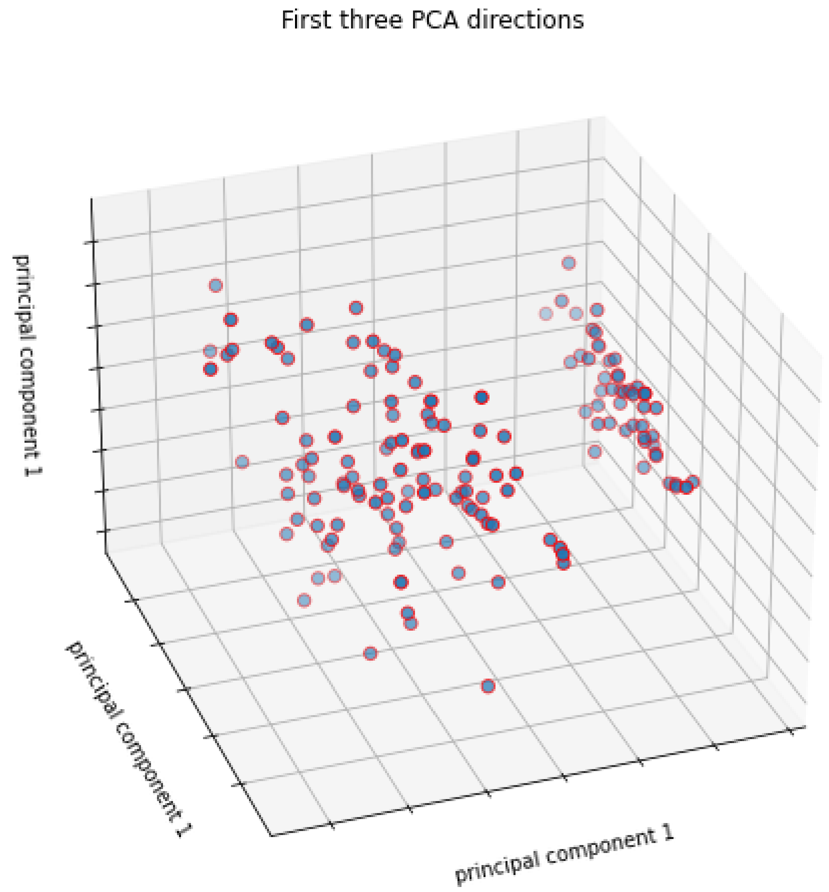
```
[-2.07535759e+00 -6.91917347e-01 -5.65590082e-02]
[-2.38125822e+00  1.15063259e+00  6.21019035e-02]
[-2.39819169e+00 -3.62390765e-01  1.46855632e-01]
[-2.22678121e+00  1.02548255e+00 -1.76645302e-01]
[-2.20595417e+00  3.22378453e-02 -1.46593527e-01]
[ 1.10399365e+00  8.63112446e-01 -6.85555108e-01]
[ 7.32481440e-01  5.98635573e-01 -9.40668020e-02]
[ 1.24210951e+00  6.14822450e-01 -5.54846534e-01]
[ 3.97307283e-01 -1.75816895e+00 -1.85694824e-02]
[ 1.07259395e+00 -2.11757903e-01 -3.97447438e-01]
[ 3.84458146e-01 -5.91062469e-01  1.26797690e-01]
[ 7.48715076e-01  7.78698611e-01  1.48656023e-01]
[-4.97863388e-01 -1.84886877e+00  2.55555250e-01]
[ 9.26222368e-01  3.03308268e-02 -5.95459889e-01]
[ 4.96802558e-03 -1.02940111e+00  5.42867049e-01]
[-1.24697461e-01 -2.65806268e+00 -3.98134482e-02]
[ 4.38730118e-01 -5.88812850e-02  2.06703491e-01]
[ 5.51633981e-01 -1.77258156e+00 -7.61380223e-01]
[ 7.17165066e-01 -1.85434315e-01 -6.72998424e-02]
[-3.72583830e-02 -4.32795099e-01  1.98061449e-01]
[ 8.75890536e-01  5.09998151e-01 -5.03505832e-01]
[ 3.48006402e-01 -1.90621647e-01  4.92831518e-01]
[ 1.53392545e-01 -7.90725456e-01 -2.98604516e-01]
[ 1.21530321e+00 -1.63335564e+00 -4.79409914e-01]
[ 1.56941176e-01 -1.30310327e+00 -1.68586746e-01]
[ 7.38256104e-01  4.02470382e-01  6.16772626e-01]
[ 4.72369682e-01 -4.16608222e-01 -2.62718283e-01]
[ 1.22798821e+00 -9.40914793e-01 -3.66704859e-01]
[ 6.29381045e-01 -4.16811643e-01 -2.89962474e-01]
[ 7.00472799e-01 -6.34939277e-02 -4.44767559e-01]
[ 8.73536987e-01  2.50708611e-01 -4.72148886e-01]
[ 1.25422219e+00 -8.26200998e-02 -7.26843529e-01]
[ 1.35823985e+00  3.28820266e-01 -2.61458074e-01]
[ 6.62126138e-01 -2.24346071e-01  8.73681069e-02]
[-4.72815133e-02 -1.05721241e+00 -3.15319195e-01]
[ 1.21534209e-01 -1.56359238e+00 -1.45241738e-01]
[ 1.41182261e-02 -1.57339235e+00 -2.36581428e-01]
[ 2.36010837e-01 -7.75923784e-01 -1.47972885e-01]
[ 1.05669143e+00 -6.36901284e-01  1.06753234e-01]
[ 2.21417088e-01 -2.80847693e-01  6.67559660e-01]
[ 4.31783161e-01  8.55136920e-01  4.50731487e-01]
[ 1.04941336e+00  5.22197265e-01 -3.96142266e-01]
[ 1.03587821e+00 -1.39246648e+00 -6.85434303e-01]
[ 6.70675999e-02 -2.12620735e-01  2.94128262e-01]
[ 2.75425066e-01 -1.32981591e+00  9.34447685e-02]
[ 2.72335066e-01 -1.11944152e+00  9.81718909e-02]
[ 6.23170540e-01  2.75426333e-02 -1.93046544e-02]
[ 3.30005364e-01 -9.88900732e-01 -1.95968073e-01]
[-3.73627623e-01 -2.01793227e+00  1.12184053e-01]
[ 2.82944343e-01 -8.53950717e-01  1.34118823e-01]
[ 8.90531103e-02 -1.74908548e-01  1.31448375e-01]
[ 2.24356783e-01 -3.80484659e-01  1.58769003e-01]
[ 5.73883486e-01 -1.53719974e-01 -2.70039416e-01]
[-4.57012873e-01 -1.53946451e+00  1.96126173e-01]
[ 2.52244473e-01 -5.95860746e-01  9.47499397e-02]
[ 1.84767259e+00  8.71696662e-01  1.00276099e+00]
[ 1.15318981e+00 -7.01326114e-01  5.31464635e-01]
[ 2.20634950e+00  5.54470105e-01 -2.05495910e-01]
[ 1.43868540e+00 -5.00105223e-02  1.63390464e-01]
[ 1.86789070e+00  2.91192802e-01  3.94004333e-01]
[ 2.75419671e+00  7.88432206e-01 -5.86232704e-01]
[ 3.58374475e-01 -1.56009458e+00  9.90999895e-01]
[ 2.30300590e+00  4.09516695e-01 -6.54166687e-01]
[ 2.00173530e+00 -7.23865359e-01 -3.94070448e-01]
[ 2.26755460e+00  1.92144299e+00  3.92517658e-01]
[ 1.36590943e+00  6.93948040e-01  2.83279516e-01]
[ 1.59906459e+00 -4.28248836e-01  2.33040821e-02]
[ 1.88425185e+00  4.14332758e-01  2.45485540e-02]
[ 1.25308651e+00 -1.16739134e+00  5.82130271e-01]
```

```
[ 1.46406152e+00 -4.44147569e-01  1.00411052e+00]
[ 1.59180930e+00  6.77035372e-01  6.36650721e-01]
[ 1.47128019e+00  2.53192472e-01  3.66575092e-02]
[ 2.43737848e+00  2.55675734e+00 -1.34200082e-01]
[ 3.30914118e+00 -2.36132010e-03 -7.06933959e-01]
[ 1.25398099e+00 -1.71758384e+00 -2.64622084e-01]
[ 2.04049626e+00  9.07398765e-01  2.31878114e-01]
[ 9.73915114e-01 -5.71174376e-01  8.29503781e-01]
[ 2.89806444e+00  3.97791359e-01 -8.60926842e-01]
[ 1.32919369e+00 -4.86760542e-01 -4.70734933e-03]
[ 1.70424071e+00  1.01414842e+00  2.95957877e-01]
[ 1.95772766e+00  1.00333452e+00 -4.22817052e-01]
[ 1.17190451e+00 -3.18896617e-01  1.30651910e-01]
[ 1.01978105e+00  6.55429631e-02  3.38042170e-01]
[ 1.78600886e+00 -1.93272800e-01  2.70002526e-01]
[ 1.86477791e+00  5.55381532e-01 -7.17510683e-01]
[ 2.43549739e+00  2.46654468e-01 -7.30234006e-01]
[ 2.31608241e+00  2.62618387e+00 -4.99619543e-01]
[ 1.86037143e+00 -1.84672394e-01  3.53330279e-01]
[ 1.11127173e+00 -2.95986102e-01 -1.82659608e-01]
[ 1.19746916e+00 -8.17167742e-01 -1.63213782e-01]
[ 2.80094940e+00  8.44748194e-01 -5.47000957e-01]
[ 1.58015525e+00  1.07247450e+00  9.43392608e-01]
[ 1.34704442e+00  4.22255966e-01  1.80028706e-01]
[ 9.23432978e-01  1.92303705e-02  4.17394303e-01]
[ 1.85355198e+00  6.72422729e-01 -1.48203294e-02]
[ 2.01615720e+00  6.10397038e-01  4.25914947e-01]
[ 1.90311686e+00  6.86024832e-01  1.27799364e-01]
[ 1.15318981e+00 -7.01326114e-01  5.31464635e-01]
[ 2.04330844e+00  8.64684880e-01  3.35266061e-01]
[ 2.00169097e+00  1.04855005e+00  6.29268888e-01]
[ 1.87052207e+00  3.82821838e-01  2.54532319e-01]
[ 1.55849189e+00 -9.05313601e-01 -2.53819099e-02]
[ 1.52084506e+00  2.66794575e-01  1.79277203e-01]
[ 1.37639119e+00  1.01636193e+00  9.31405052e-01]
[ 9.59298576e-01 -2.22839447e-02  5.28794187e-01]]
```

Data Visualisation

In [11]:

```
fig = plt.figure(1, figsize=(8, 6))
ax = Axes3D(fig, elev=-150, azim=110)
X_reduced = datapoints
ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], cmap=plt.cm.Set1, edge
ax.set_title("First three PCA directions")
ax.set_xlabel("principal component 1")
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("principal component 1")
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("principal component 1")
ax.w_zaxis.set_ticklabels([])
plt.show()
```



Algorithm Implementation

In [12]:

```
def init_mediods(X, k):
    np.random.seed(1)
    samples = np.random.choice(len(X), size=k, replace=False)
    return X[samples, :]
```

In [13]:

```
mediods_initial = init_mediods(datapoints, 3)
print(mediods_initial)
```

```
[[-2.19229151  1.88997851 -0.46948009]
 [-0.45701287 -1.53946451  0.19612617]
 [ 0.87353699  0.25070861 -0.47214889]]
```

In [15]:

```
def comp_distance(X, mediods, p):
    m = len(X)
    mediods_shape = mediods.shape

    if len(mediods_shape) == 1:
        mediods = mediods.reshape((1, len(mediods)))

    k = len(mediods)

    S = np.empty((m, k))

    for i in range(m):
        d_i = np.linalg.norm(X[i, :]-mediods, ord=p, axis=1)
        S[i, :] = d_i**p

    return S
```

In [16]:

```
S = comp_distance(datapoints, mediods_initial, 2)
print(S)
```

```
[[ 2.04221808  7.55104262 10.03520454]
 [ 6.54885847  3.6157953   9.64239437]
 [ 5.17953221  5.16341479 11.10540023]
 [ 6.41346585  4.35103305 11.10644807]
 [ 1.75633541  8.66505699 11.06613962]
 [ 0.4029488  11.98227445 10.52782982]
 [ 4.01877701  6.58134222 11.71159711]
 [ 2.84878381  6.42844465  9.8075811 ]
 [ 9.32155243  3.75224721 12.54224615]
 [ 5.51879198  4.38454897  9.9171106 ]
 [ 0.71443634  9.93619081  9.9390487 ]
 [ 3.34045952  6.39085369 10.58188538]
 [ 6.81696092  3.98785753 10.57848361]
 [ 8.6335934   5.1256575  14.19451894]
 [ 0.         15.21530309 12.0865173 ]
 [ 0.94846714 21.41004134 16.12952938]
 [ 0.36079023 12.40872253 11.28045773]
 [ 2.07812819  7.27693545  9.64379046]
 [ 0.30924222 11.21080312  9.0633045 ]
 [ 0.92587132 10.82539054 11.52130242]
 [ 2.21017539  6.37976291  7.80892905]
 [ 1.28133164  9.26487903 10.37328339]
 [ 2.95325024  9.50634848 14.02049133]
 [ 3.5778284   4.59323701  7.5404421 ]
 [ 3.33901483  6.03795401  9.98435082]
 [ 6.3239138   3.3626048   8.77939386]
 [ 2.97228799  5.81476259  8.87774025]
 [ 1.86264605  7.46037643  9.4162733 ]
 [ 2.45724533  6.56617283  9.13341405]
 [ 5.16426708  4.79669466 10.49404522]
 [ 5.79907999  4.03161053  9.81204039]
 [ 2.2587052   6.04256146  7.4002037 ]
 [ 0.44810132 15.99157625 14.88598368]
 [ 0.2998191  17.87455545 14.86324124]
 [ 5.51879198  4.38454897  9.9171106 ]
 [ 4.36607123  5.08527797  9.76864808]
 [ 1.47404387  7.92425008  8.70307658]
 [ 5.51879198  4.38454897  9.9171106 ]
 [ 8.19446623  4.34826722 12.69001173]
 [ 2.6414105   6.42712883  9.35954519]
 [ 2.29124784  7.40114925 10.29626932]
 [17.96711383  2.77527322 14.26158647]
 [ 6.24016645  5.60587652 12.88802225]
 [ 2.60631341  6.43470797  8.73327243]
 [ 1.04145908 10.16235437 10.41477949]
 [ 6.85036359  3.40122563  9.75723789]
 [ 0.86492023 10.95730507 11.6889789 ]
 [ 5.49543197  5.15610538 11.46326596]
 [ 0.83429502  9.84999199 10.29956997]
 [ 3.55564279  5.64650074  9.63698155]
 [11.96663819  8.98647928  0.47369095]
 [10.36279853  6.07058065  0.2838959 ]
 [13.42842074  8.09192917  0.27526349]
 [20.21832217  0.82378878  4.46811805]
 [15.08196171  4.45483149  0.25907926]
 [13.15075033  1.61234635  1.30651366]
 [10.26655501  6.82991356  0.67975268]
 [17.37574192  0.10093158  6.81851748]
 [13.19928922  5.00420562  0.06654772]
 [14.37557359  0.59382025  3.42335029]
 [25.14423724  1.4173629   9.64433636]
 [11.17755355  2.99459407  0.74574347]
 [21.0286791   1.98853054  4.28097945]
 [12.93402492  3.28148487  0.37857558]
 [10.48505673  1.40091476  1.74590742]
```

```
 [11.31924461  6.46641364  0.06821986]
 [11.70805413  2.55546716  1.40214197]
 [12.71760591  1.1779634   1.63331058]
 [24.02568394  3.26180563  3.66655499]
 [15.80520228  0.56582172  3.01999089]
 [11.98073472  5.37672249  1.22708261]
 [12.46351207  2.33509641  0.65010814]
 [19.72283303  3.5142692   1.55672045]
 [13.3153434   2.67688338  0.5383873 ]
 [12.18475062  3.92900702  0.12942419]
 [12.0865173   5.42167429  0.        ]
 [15.8358379   5.90259415  0.32089861]
 [15.08676114  6.99501379  0.28542891]
 [12.92814846  2.99383695  0.58343076]
 [13.31076781  0.66202324  2.58315967]
 [17.38607187  0.45183093  3.96406455]
 [16.91742347  0.41035148  4.12143697]
 [13.10705422  1.18168046  1.56550375]
 [17.27305647  3.11390861  1.15652454]
 [11.83133501  2.26663304  2.00674804]
 [ 8.80345435  6.58889827  1.41218821]
 [12.3848544   6.87055116  0.11041559]
 [21.24216111  3.02748108  2.77186973]
 [10.1087249   2.04477916  1.45224762]
 [16.77358564  0.59096134  3.17569182]
 [15.45322189  0.71796277  2.56402093]
 [11.59815188  3.66871803  0.31755431]
 [14.72473604  1.07625608  1.90833379]
 [18.91763816  0.24293077  7.04359597]
 [14.02027179  1.02131075  1.93663244]
 [ 9.82940687  2.16438425  1.16089457]
 [11.38988891  1.80889446  1.21789726]
 [11.86820419  3.20034558  0.29420294]
 [15.21530309  0.          5.42167429]
 [12.47350852  1.40371119  1.42405834]
 [19.52570168 11.77593302  3.50992547]
 [18.90899526  3.40768053  1.99181592]
 [21.20131305 11.63936158  1.93976404]
 [17.34807593  5.81321676  0.81373499]
 [19.78680069  8.79563859  1.74059968]
 [25.69478112 16.34305551  3.83904277]
 [20.54190313  1.29710636  5.68520498]
 [22.4335751  12.1392293   2.10173185]
 [24.42772726  7.0589766   2.2287221 ]
 [20.63425744 19.43971826  5.4822867 ]
 [14.65792984  8.31877304  1.00956388]
 [19.99139532  5.49212226  1.23284715]
 [19.03980048  9.32828316  1.29502575]
 [22.32402447  3.21187752  3.26656984]
 [20.98853137  5.54308475  3.01088619]
 [17.01417539  9.30460591  1.92710619]
 [16.3570014   6.95736344  0.61618705]
 [21.99085092 25.26565021  7.87767036]
 [33.90309603 17.36212019  6.05133622]
 [24.93326707  3.17151544  4.0619802 ]
 [19.37385859 12.22596995  2.28868992]
 [17.76949703  3.38630788  2.37986708]
 [28.2915767  16.12686513  4.27149302]
 [18.26576025  4.33905364  0.96998537]
 [16.53593695 11.20192243  1.86289702]
 [18.01097413 12.67988945  1.7443488 ]
 [16.5571026   4.14744469  0.77684209]
 [14.29806742  4.77710944  0.71208319]
 [20.71364441  6.84883633  1.58051313]
 [18.30248056 10.6138247   1.13558657]
 [24.18493662 12.41497981  2.50634467]
 [20.86834212 25.52674598  7.72457464]
 [21.40527015  7.23044479  1.84481468]
 [15.77423733  4.1492338   0.43919696]
```

```
 [18.91291725  3.3881486   1.34073285]
 [26.03097071 16.85102634  4.07340448]
 [16.89587663 11.53068645  3.17836626]
 [15.10297     7.10322909  0.6789734 ]
 [13.99398388  4.38412006  0.84735887]
 [18.05800715 10.27565354  1.3474216 ]
 [20.15010157 10.79127777  2.24147536]
 [18.57861691 10.52768371  1.60947283]
 [18.90899526  3.40768053  1.99181592]
 [19.63915032 12.05090086  2.39725101]
 [19.50474019 12.93065653  3.12240347]
 [19.3021694   9.116015    1.53949874]
 [22.07925731  4.51347266  2.00515131]
 [16.84299533  7.17477779  0.84362245]
 [15.46118134 10.43425427  2.80905101]
 [14.5858193   4.41844299  1.08376702]]
```

In [17]:

```
def assign_labels(S):
    return np.argmin(S, axis=1)
```

In [18]:

```
labels = assign_labels(S)
print(labels)
```

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

In [20]:

```
def update_mediods(X, mediods, p):
    S = comp_distance(datapoints, mediods, p)
    labels = assign_labels(S)

    out_mediods = mediods

    for i in set(labels):
        avg_dissimilarity = np.sum(comp_distance(datapoints, mediods[i], p))

        cluster_points = datapoints[labels==i]

        for datap in cluster_points:
                new_medoid = datap
                new_dissimilarity= np.sum(comp_distance(datapoints, datap, p))

                if new_dissimilarity < avg_dissimilarity :
                    avg_dissimilarity = new_dissimilarity

                    out_mediods[i] = datap

    return out_mediods
```

In [21]:

```
def has_converged(old_mediods, mediods):
      return set([tuple(x) for x in old_mediods]) == set([tuple(x) for x in mediods]
```

In [22]:

```
def kmediods(X, k, p, starting_mediods=None, max_steps=np.inf):
    if starting_mediods is None:
        mediods = init_mediods(X, k)
    else:
        mediods = starting_mediods

    converged = False
```

```
    labels = np.zeros(len(X))

    i = 1

    while (not converged) and (i<=max_steps):
        old_mediods = mediods.copy()

        S = comp_distance(X, mediods, p)

        labels = assign_labels(S)

        mediods = update_mediods(X, mediods, p)

        converged = has_converged(old_mediods, mediods)
        i+=1

    return (mediods, labels)
```

Results

In [23]:

```
results = kmediods(datapoints, 3, 2)
final_mediods = results[0]
data['clusters'] = results[1]
```

In [24]:

```
print(final_mediods)
```

```
[[-1.82041156  0.10675079  0.04006147]
 [-0.03725838 -0.4327951   0.19806145]
 [ 0.08905311 -0.17490855  0.13144838]]
```

In [25]:

```
print(data['clusters'])
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: clusters, Length: 150, dtype: int64
```

Analysis

- Given two Numpy arrays of {0, 1} labels, returns a new boolean array indicating at which locations the input arrays have the same label (i.e., the corresponding entry is True). This function can consider "inexact" matches. That is, if `exact` is False, then the function will assume the {0, 1} labels may be regarded as the same up to a swapping of the labels. This feature allows a = [0, 0, 1, 1, 0, 1, 1] b = [1, 1, 0, 0, 1, 0, 0] to be regarded as equal. (That is, use `exact=False` when you only care about "relative" labeling.)

In [26]:

```
def mark_matches(a, b, exact=False):
```

```
    assert a.shape == b.shape
    a_int = a.astype(dtype=int)
    b_int = b.astype(dtype=int)
    all_axes = tuple(range(len(a.shape)))
    assert ((a_int == 0) | (a_int == 1) | (a_int == 2)).all()
    assert ((b_int == 0) | (b_int == 1) | (b_int == 2)).all()

    exact_matches = (a_int == b_int)
    if exact:
        return exact_matches

    assert exact == False
    num_exact_matches = np.sum(exact_matches)
    if (2*num_exact_matches) >= np.prod (a.shape):
        return exact_matches
    return exact_matches == False # Invert
```

In [27]:

```
def count_matches(a, b, exact=False):
    matches = mark_matches(a, b, exact=exact)
    return np.sum(matches)
```

In [28]:

```
n_matches = count_matches(labels, data['clusters'])
print(n_matches, "matches out of",len(data), "data points", "(~ {:.1f}%)".format(100
```

```
130 matches out of 150 data points (~ 86.7%)
```

In [29]:

```
Comparison = pd.DataFrame({'Actual':y_data,'Predicted':results[1]})
Comparison
```

Out[29]:

	Actual	Predicted
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
145	2	2
146	2	2
147	2	2
148	2	2
149	2	2

150 rows × 2 columns

In [ ]: