# Exercise 9 – Cluster Analysis using Kmeans and Kmediods

S.DEIVANAYAKI (21BCS003)

1. **Python Program to build cluster using KMeans with random data points without API.**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.DataFrame({'X': np.random.choice(range(100),100),'Y':
np.random.choice((100),100)})

k=int(input("Specify the number of cluster : "))
cluster=[]
for i in range(k):
    ind=int(input("Specify the cluster index (0-99) : "))
    cluster.append(data.iloc[ind,:])
cluster
```

```
Specify the number of cluster :  3
Specify the cluster index (0-99) :  6
Specify the cluster index (0-99) :  87
Specify the cluster index (0-99) :  45


[X    18
 Y    82
 Name: 6, dtype: int32,
 X    29
 Y    15
 Name: 87, dtype: int32,
 X    15
 Y    42
 Name: 45, dtype: int32]
```

```
diff=[]
itr=0
while(1):
    df=[]
    for i in range(k):
        df.append(np.sum((data-cluster[i])**2,axis=1) ** 0.5)
    df=pd.DataFrame(df).transpose()
    cg=np.argmin(df,axis=1)
    diff.append(cg)
    for i in range(k):
        print("Points belongs to Cluster",i)
        print(data.iloc[cg==i,])
    cluster = [[0, 0] for _ in range(k)]
    for i in range(len(cg)):
        cluster[cg[i]][0]+=data.iloc[i,0]
        cluster[cg[i]][1]+=data.iloc[i,1]
    for i in range(k):
        cluster[i]=(np.array(cluster[i])/cg.tolist().count(i)).tolist()
    print("New Cluster Points :",cluster)
    if(itr!=0 and sum(diff[itr-1]==diff[itr])==len(data)):
        print("No Changes in Cluster Group.. Hence Iteration Stops!!")
        break
    itr+=1
```

```
Points belongs to Cluster 0
     X   Y
0   66  65
2   29  95
3   25  71
6   18  82
Points belongs to Cluster 1
     X   Y
1   94  19
5   82  33
7   51   3
10  98  76
Points belongs to Cluster 2
     X   Y
4   35  29
9   22  17
12   7  33
18  33  41
22   3  11
New Cluster Points : [[42.2222222222222, 77.68518518518519],
[75.15384615384616, 29.807692307692307], [19.6, 28.7]]
Points belongs to Cluster 0
     X   Y
0   66  65
2   29  95
3   25  71
6   18  8
Points belongs to Cluster 1
     X   Y
1   94  19
5   82  33
7   51   3
10  98  76
17  68   4
Points belongs to Cluster 2
     X   Y
4   35  29
```

```
9   22  17
12   7  33
18  33  41
New Cluster Points : [[42.22222222222222, 77.68518518518519],
[75.15384615384616, 29.807692307692307], [19.6, 28.7]]
No Changes in Cluster Group.. Hence Iteration Stops!!
```

2. **Python Program to build cluster using KMediods with random data points without API.**

```
k=int(input("Specify the number of cluster : "))
cluster=[]
for i in range(k):
    ind=int(input("Specify the cluster index (0-99) : "))
    cluster.append(data.iloc[ind,:].tolist())
cluster
```

```
Specify the number of cluster :  3
Specify the cluster index (0-99) :  12
Specify the cluster index (0-99) :  13
Specify the cluster index (0-99) :  14
```

```
[[7, 33], [46, 75], [55, 67]]
```

```
totalcost=9999
while(1):
    print("Cluster Points :\n",cluster)
    df=[]
    for i in range(k):
        df.append(np.sum(abs(data-cluster[i]),axis=1))
    df=pd.DataFrame(df).transpose()
    cg=np.argmin(df,axis=1)
    for i in range(k):
            print("Points belongs to Cluster ",i)
            print(data.iloc[cg==i,])
    cost=[]
    for i in range(k):
        cost.append(sum(df.iloc[cg==i,i]))
    print("Cluster-wise Cost :",cost)
    temp=sum(cost)
    mini=99999
    minind=-1;
    for i in range(k):
        if(len(df.iloc[cg==i,i])>1):
            df1=(df.iloc[cg==i,i])
            ind=np.array(df.iloc[cg==i,i]!=0)
            df1=df1.iloc[ind,]
            if(len(df1)!=0 and mini>min(df1)):
                mini=min(df1)
                minind=i
                a=df1[df1==min(df1)].index
    cluster[minind]=(data.iloc[a,:].values)
    print("Current Iteration Total Cost :",temp)
    if(totalcost>temp and temp!=0):
        totalcost=temp
    else:
        print("Previous Iteration Cost is Efficient!!, Hence No Change in
Cost!!, Cost is",totalcost)
        break;
```

```
Cluster Points :
 [[7, 33], [46, 75], [55, 67]]
Points belongs to Cluster  0
     X   Y
4   35  29
9   22  17
12   7  33
18  33  41
Points belongs to Cluster  1
     X   Y
2   29  95
3   25  71
6   18  82
8   17  86
Points belongs to Cluster  2
     X   Y
0   66  65
1   94  19
5   82  33
7   51   3
Cluster-wise Cost : [520, 951, 2053]
Current Iteration Total Cost : 3524
Cluster Points :
 [[7, 33], array([[47, 74]]), [55, 67]]
Points belongs to Cluster  0
     X   Y
4   35  29
9   22  17
12   7  33
18  33  41
Points belongs to Cluster  1
     X   Y
2   29  95
3   25  71
6   18  82
8   17  86
Points belongs to Cluster  2
     X   Y
0   66  65
```

```
1   94  19
5   82  33
7   51   3
Cluster-wise Cost : [520, 993, 2053]
Current Iteration Total Cost : 3566
Previous Iteration Cost is Efficient!!, Hence No Change in Cost!!, Cost is
3524
```

3. **Python Program to build cluster using KMeans with random data points with API.**

```
data = pd.read_csv("C:/Users/DELL/Downloads/seeds_dataset.txt",
names=['area', 'perimeter', 'compactness', 'length_of_kernel',
'width_of_kernel', 'asymmetry_coefficient', 'length_of_kernel_groove',
'target'],delimiter='\t')
```

```
X=data.iloc[:,0:7]
```

```
n=int(input("Specify the number of clusters : "))
```

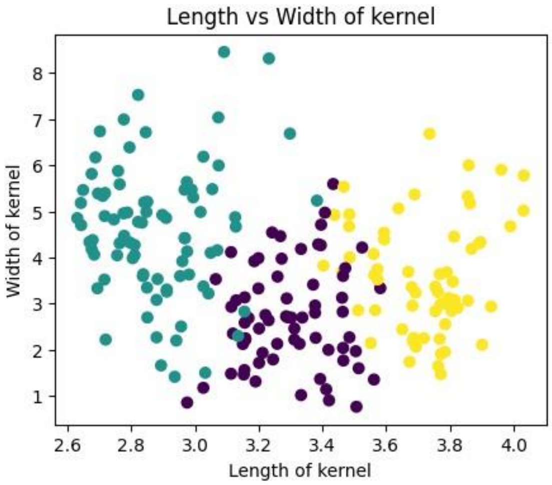
```
Specify the number of clusters :  3
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=n)
kmeans.fit(X)
kmeans_labels = kmeans.labels_
kmeans_labels
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2,
       2, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1,
       2, 2, 2, 2, 1, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
plt.figure(figsize=(5,4))
plt.scatter(X.iloc[:, 4], X.iloc[:, 5], c=kmeans_labels)
plt.title('Length vs Width of kernel')
plt.xlabel('Length of kernel')
plt.ylabel('Width of kernel')
```

```
Text(0, 0.5, 'Width of kernel')
```



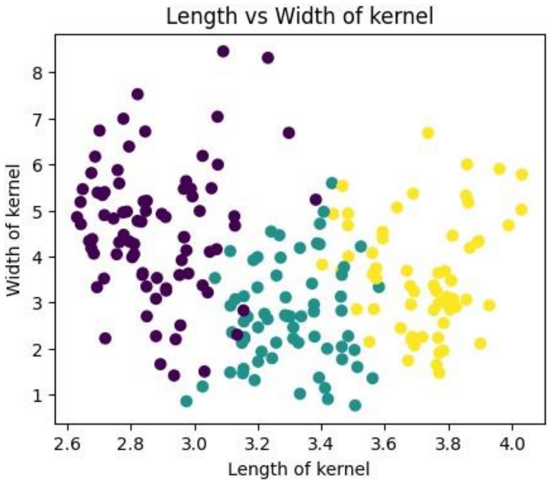
4. **Python Program to build cluster using KMediods with random data points with API.**

```
from sklearn_extra.cluster import KMedoids
kmedoids = KMedoids(n_clusters=3)
kmedoids.fit(X)
kmedoids_labels = kmedoids.labels_
kmedoids_labels
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 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, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2,
       1, 1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

```
plt.figure(figsize=(5,4))
plt.scatter(X.iloc[:, 4], X.iloc[:, 5], c=kmedoids_labels)
plt.title('Length vs Width of kernel')
plt.xlabel('Length of kernel')
plt.ylabel('Width of kernel')
```

```
Text(0, 0.5, 'Width of kernel')
```



5. **Python Program to apply kmeans on the simple digit dataset.**

```
from sklearn.datasets import load_digits
ds=load_digits().data
ds
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

```
from sklearn.preprocessing import scale
scaled_ds = scale(ds)
scaled_ds
```

```
array([[ 0.        , -0.33501649, -0.04308102, ..., -1.14664746,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  0.54856067,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  1.56568555,
         1.6951369 , -0.19600752],
       ...,
       [ 0.        , -0.33501649, -0.88456568, ..., -0.12952258,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -0.67419451, ...,  0.8876023 ,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649,  1.00877481, ...,  0.8876023 ,
        -0.26113572, -0.19600752]])
```

```
kmeans = KMeans(n_clusters=10)
kmeans.fit(scaled_ds)
```

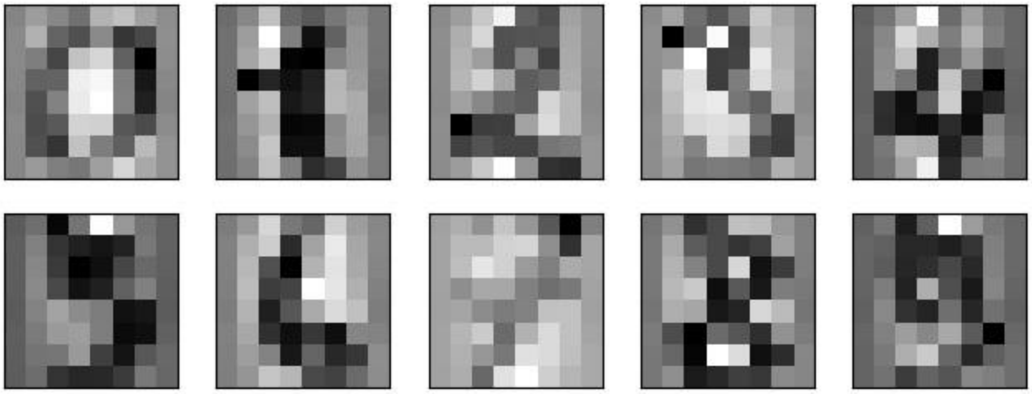
```
y_pred=kmeans.predict(ds)
y_pred
```

```
array([7, 5, 5, ..., 5, 4, 5])
```

```
from sklearn.metrics import silhouette_score
print("Silhoutte Score :",silhouette_score(ds,y_pred))
```

```
Silhoutte Score : 0.18175957251641026
```

```
fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = scaled_ds[:10].reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
   axi.set(xticks=[], yticks=[])
   axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)
```

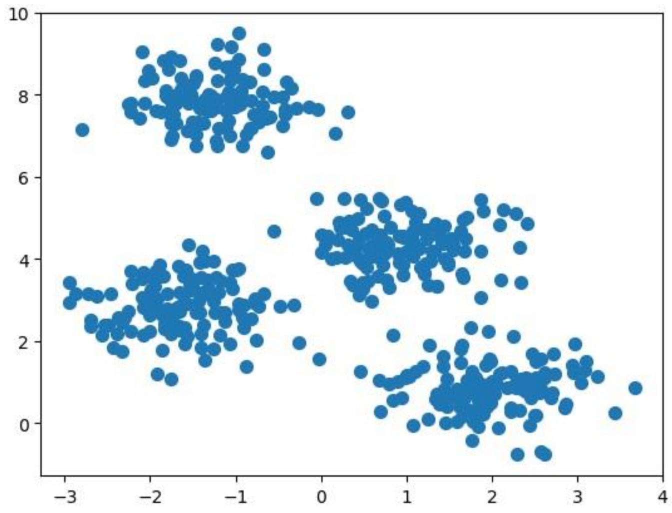


```
ds
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

6. **Python Program to plot the cluster centers using Kmeans for 4 distinct blobs.**

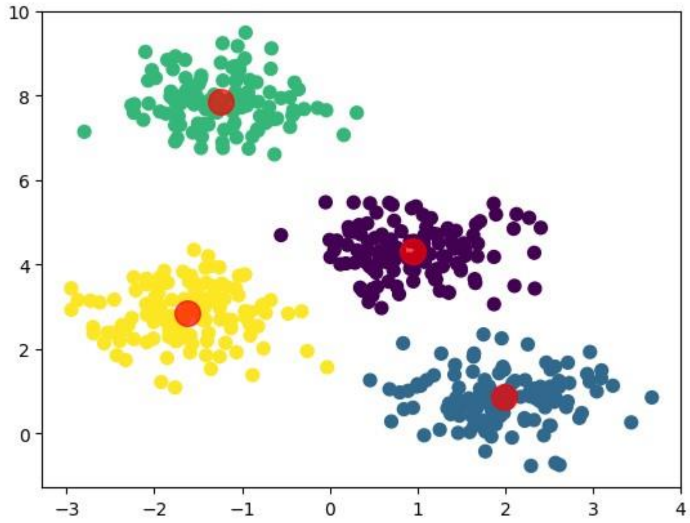
```
from sklearn.datasets import make_blobs
X, y_true =
make_blobs(n_samples=500,centers=4,cluster_std=0.60,random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```



```
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_pred = kmeans.predict(X)
y_pred[:20]
```

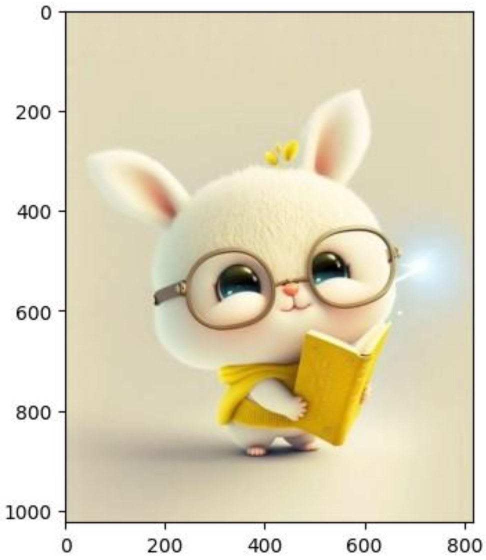
```
array([1, 1, 0, 0, 1, 3, 3, 0, 0, 2, 3, 1, 3, 2, 0, 0, 1, 0, 2, 0])
```

```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.7);
```



7. **Python Program to visualize, reshape and rescale the colors in a image using Kmeans.**

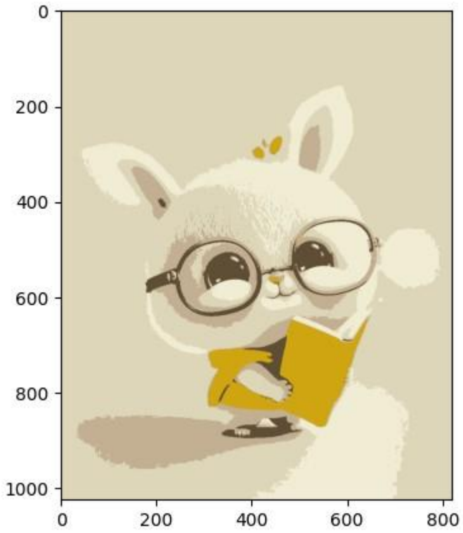
```
from PIL import Image
image = Image.open("C:/Users/DELL/Pictures/WALPAPER/1685458846968.jpg")
plt.imshow(image)
```



```
imgarr = np.array(image)
reimg = np.reshape(image, (-1, imgarr.shape[-1]))

resimg = reimg / 255.0

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(resimg)
centers = kmeans.cluster_centers_
labels = kmeans.labels_newimg =
centers[labels].reshape(imgarr.shape)plt.figure(figsize=(8, 5))
plt.imshow((newimg*255).astype(np.uint8))
```


0
200
400
600
800
1000
0
200
400
600
800