Hands-on Exercise for CLUS Module

0. Setting up necessary packages and creating data

In [2]:

```
!pip install --user scikit-learn --upgrade
```

```
Requirement already up-to-date: scikit-learn in /users/PES0801/dogipakr/.l
ocal/lib/python3.6/site-packages
Requirement already up-to-date: joblib>=0.11 in /users/PES0801/dogipakr/.l
ocal/lib/python3.6/site-packages (from scikit-learn)
Requirement already up-to-date: numpy>=1.11.0 in /users/PES0801/dogipakr/.
local/lib/python3.6/site-packages (from scikit-learn)
Requirement already up-to-date: scipy>=0.17.0 in /users/PES0801/dogipakr/.
local/lib/python3.6/site-packages (from scikit-learn)
You are using pip version 9.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

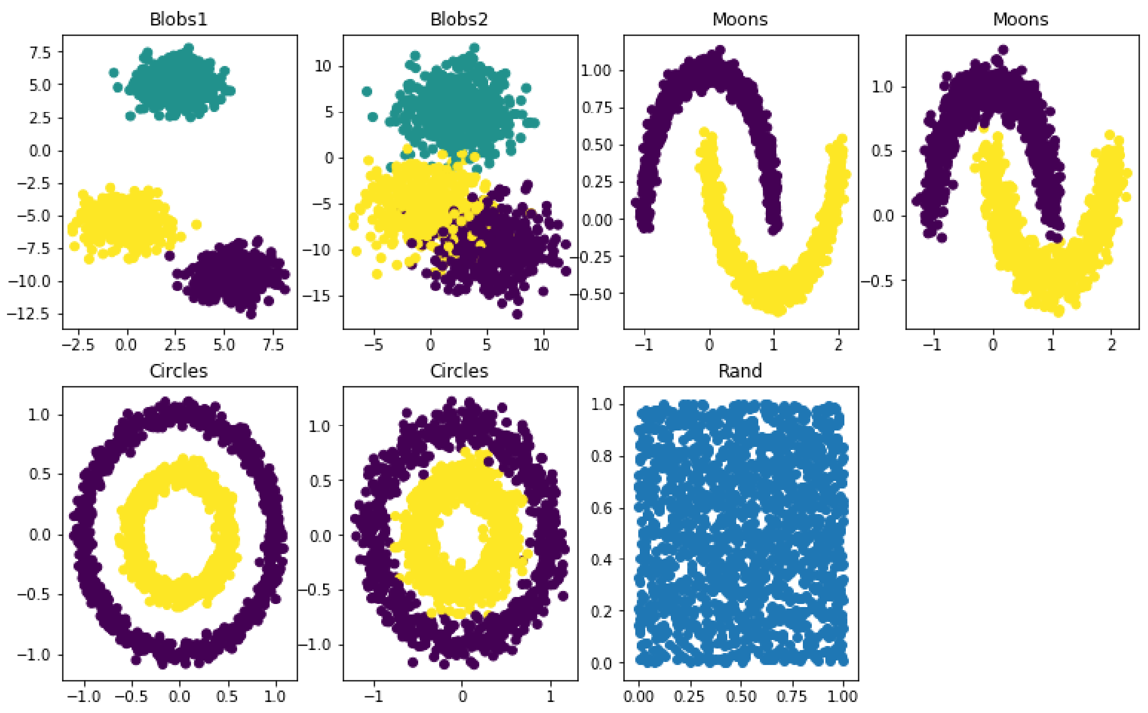
Import necessary packages

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns

from sklearn import datasets

# importing clustering algorithms
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import SpectralClustering

from sklearn.metrics import silhouette_samples
```

In [4]:

```
n_samples = 1500
random_state = 10

Blobs1_X, Blobs1_y = datasets.make_blobs(n_samples=n_samples,
                             random_state=random_state)
Blobs2_X, Blobs2_y = datasets.make_blobs(n_samples=n_samples,
                             cluster_std=[2.5, 2.5, 2.5],
                             random_state=random_state)
Moons1_X, Moons1_y = datasets.make_moons(n_samples=n_samples, noise=0.05,
                             random_state=random_state)
Moons2_X, Moons2_y = datasets.make_moons(n_samples=n_samples, noise=0.1,
                             random_state=random_state)
Circles1_X, Circles1_y = datasets.make_circles(n_samples=n_samples, factor=.5,
                                      noise=.05, random_state=random_state)
Circles2_X, Circles2_y = datasets.make_circles(n_samples=n_samples, factor=.5,
                                      noise=0.1, random_state=random_state)
Rand_X = np.random.rand(n_samples, 2);
plt.figure(figsize=(13,8))

plt.subplot(2,4,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c= Blobs1_y)
plt.title('Blobs1')

plt.subplot(2,4,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c= Blobs2_y)
plt.title('Blobs2')

plt.subplot(2,4,3)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c= Moons1_y)
plt.title('Moons')

plt.subplot(2,4,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c= Moons2_y)
plt.title('Moons')

plt.subplot(2,4,5)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c= Circles1_y)
plt.title('Circles')

plt.subplot(2,4,6)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c= Circles2_y)
plt.title('Circles')

plt.subplot(2,4,7)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand')
plt.show()
```



Code for RandIndex function

In [5]:

```
from scipy.special import comb
def rand_index(S, T):

    Spairs = comb(np.bincount(S), 2).sum()
    Tpairs = comb(np.bincount(T), 2).sum()

    A = np.c_[(S, T)]

    f_11 = sum(comb(np.bincount(A[A[:, 0] == i, 1]), 2).sum()
             for i in set(S))

    f_10 = Spairs - f_11
    f_01 = Tpairs - f_11
    f_00 = comb(len(A), 2) - f_11 - f_10 - f_01
    return (f_00 + f_11) / (f_00 + f_01 + f_10 + f_11)
```

Code for Hopkins statistic

In [6]:

```
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan
def hopkins(X):
    n=X.shape[0]#rows
    d=X.shape[1]#cols
    p=int(0.1*n)#considering 10% of points
    nbrs=NearestNeighbors(n_neighbors=1).fit(X)

    rand_X=sample(range(0,n),p)
    uj=[]
    wj=[]
    for j in range(0,p):
        u_dist,_=nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape
(1,-1),2,return_distance=True)
        uj.append(u_dist[0][1])#distances to nearest neighbors in random data
        w_dist,_=nbrs.kneighbors(X[rand_X[j]].reshape(1,-1),2,return_distance=True)
        wj.append(w_dist[0][1])#distances to nearest neighbors in real data
    H=sum(uj)/(sum(uj)+sum(wj))
    if isnan(H):
        print(uj,wj)
        H=0

    return H
```

Code for Silhouette coefficient

In [7]:

```
def silhouette(X,labels):
    n_clusters=np.size(np.unique(labels));
    sample_silhouette_values=silhouette_samples(X,labels)
    y_lower=10
    for i in range(n_clusters):
        ith_cluster_silhouette_values=sample_silhouette_values[labels==i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i=ith_cluster_silhouette_values.shape[0]
        y_upper=y_lower+size_cluster_i
        color=cm.nipy_spectral(float(i)/n_clusters)
        plt.fill_betweenx(np.arange(y_lower,y_upper),0,ith_cluster_silhouette_values,fa
cecolor=color,edgecolor=color,alpha=0.7)# Label the silhouette plots with their cluster
numbers at the middle
        plt.text(-0.05,y_lower+0.5*size_cluster_i,str(i))#Compute the new y_lower for n
ext cluster
        y_lower=y_upper+10# 10 for the 0 samples
    plt.title("Silhouette plot for the various clusters.")
    plt.xlabel("Silhouette coefficient values")
    plt.ylabel("Cluster label")
    plt.show()
```

1. K-Means clustering

Question 1a: Without running K-Means clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where k-Means is expected to work well. Support your answer by explaining your rationale.

Answer: K-means clustering will work efficiently for globular/ellipsoid shaped data as K-Means employs a greedy iterative approach to find a clustering that minimizes the SSE objective and here in the given datasets K means is expected to work well for Blobs1,Blobs2 datasets.

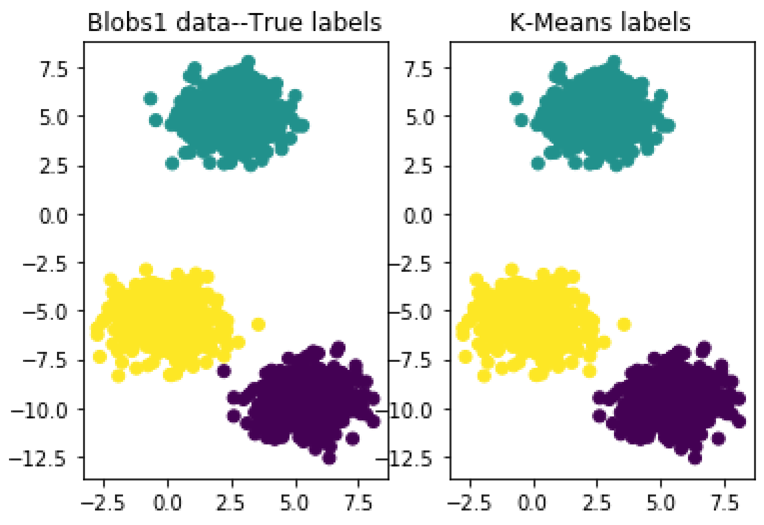
Question 1b: Without running K-Means clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where k-Means is expected to NOT work well. Support your answer by explaining your rationale.

Answer: K means clustering won't work well on the datasets: 1)with nonconvex shape clusters as two points from different clusters may be closer than two points in the same cluster . 2)it won't work well when a line joining the points in a cluster lies outside the cluster . K means won't work on Moons1_X,Moons2_X as it falls under category 2. K means won't work well on Circles1_X,Circles2_X as it falls under category1.

Question 1c: Run K-Means algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of K-means performance. Describe your rationale for your ranking.

In [8]:

```
n_clusters = 3
n_clusters1 = 2
random_state=10
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y1_pred = kmeans.fit_predict(Blobs1_X)
y2_pred = kmeans.fit_predict(Blobs2_X)
kmeans1 = KMeans(n_clusters=n_clusters1, random_state=random_state);
y3_pred = kmeans1.fit_predict(Moons1_X)
y4_pred = kmeans1.fit_predict(Moons2_X)
y5_pred = kmeans1.fit_predict(Circles1_X)
y6_pred = kmeans1.fit_predict(Circles2_X)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('K-Means labels')
plt.show()
```



In [9]:

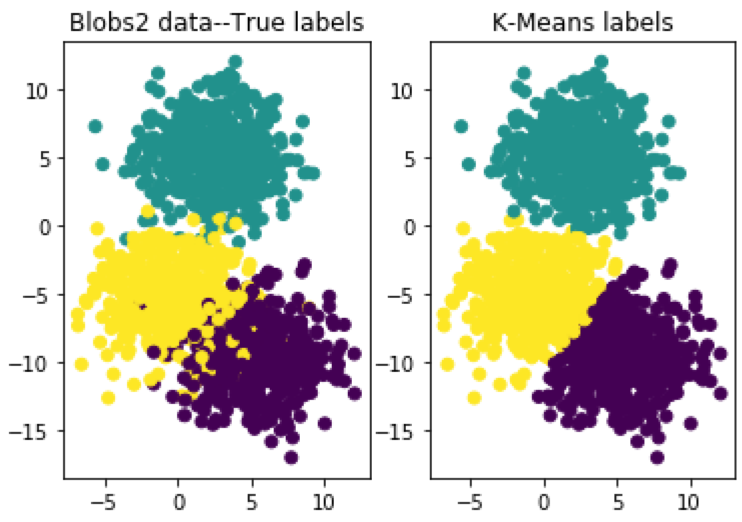
```
np.unique(y1_pred)
```

Out[9]:

```
array([0, 1, 2], dtype=int32)
```

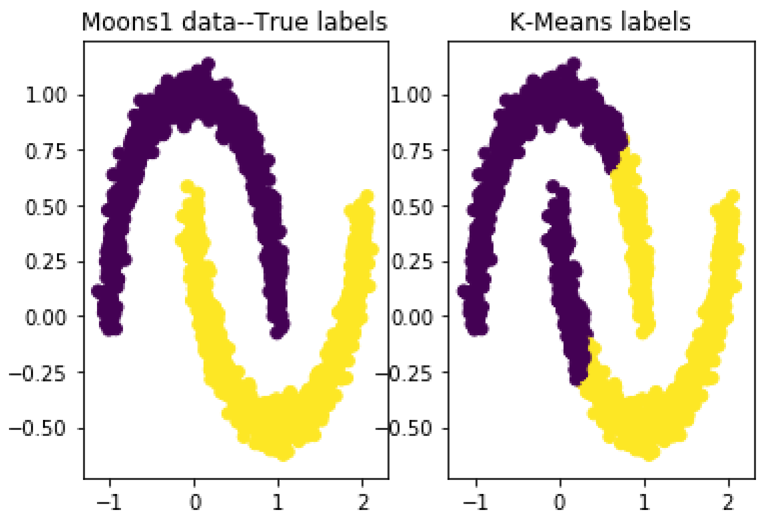
In [10]:

```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('K-Means labels')
plt.show()
```



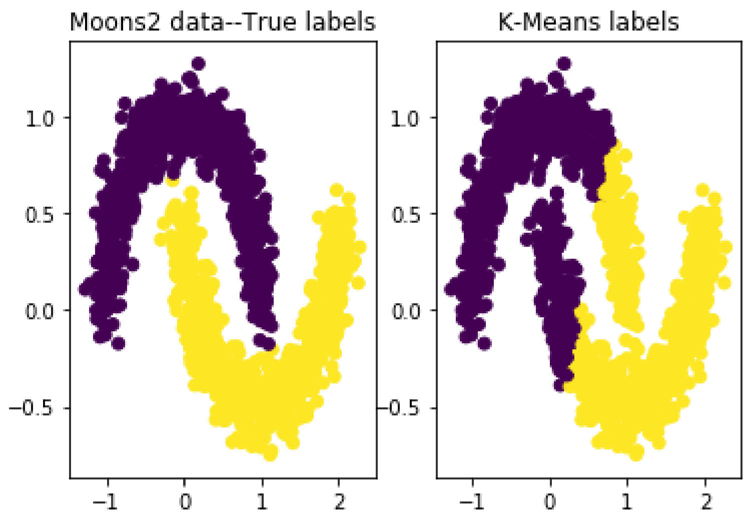
In [11]:

```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('K-Means labels')
plt.show()
```



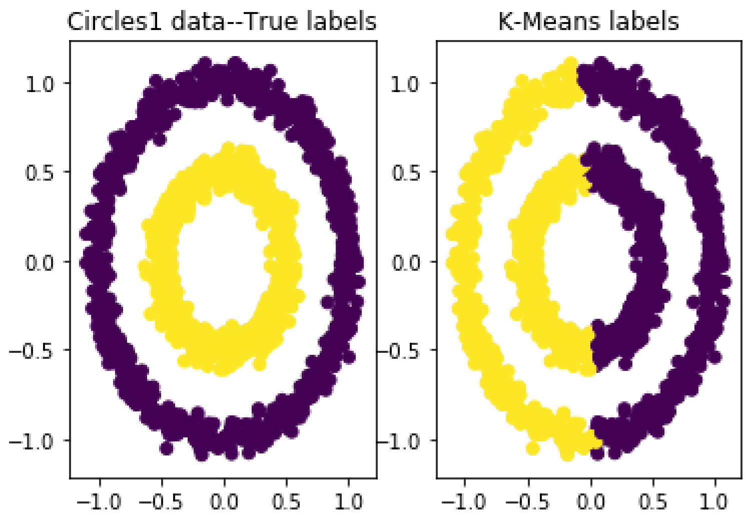
In [12]:

```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('K-Means labels')
plt.show()
```



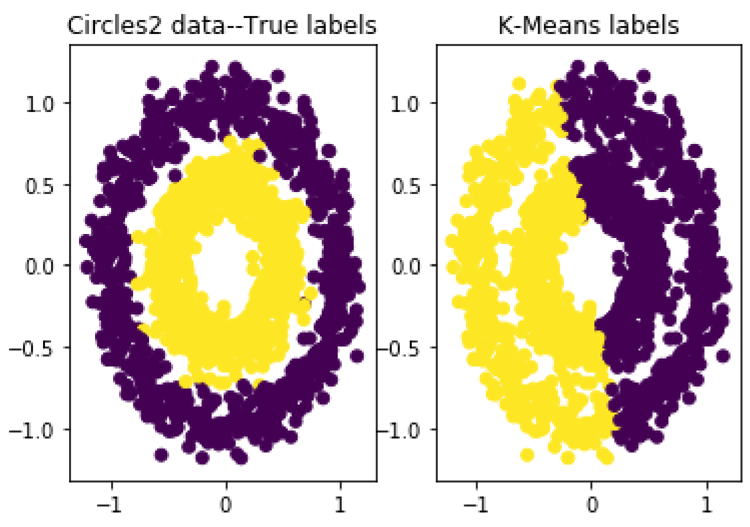
In [13]:

```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles1 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title('K-Means labels')
plt.show()
```



In [14]:

```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles2 data--True labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('K-Means labels')
plt.show()
```



Answer: Based on the visualization ,the decreasing order of K-means performance on the datasets is Blobs1_X,Blobs2_X,Moons2_X,Moons1_X,Circles1_X,Circles2_X

using K means on Blobs1_X,Blobs2_X clusters were formed exactly as of the true clusters and K means algorithm suited for globular data.

For Moons1_X,Moons2_X more than 50% of the points were clustered correctly in each cluster ,the distance between the extreme points of a particular moon is more when the compared with extreme points distance between the moons as it is distance based algorithm it misclusted the extreme points .

For Circles1_x,Circles2_X only few points were clustered correctly when compared with other datasets .

Question 1d: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using K-means. Rank the datasets in decreasing order of Rand-Index scores.

In [15]:

```
print(Blobs1_y)
```

```
[2 1 1 ... 1 1 1]
```

In [16]:

```
Rand_Blobs1=rand_index(y1_pred,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(y2_pred,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(y3_pred,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(y4_pred,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(y5_pred,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(y6_pred,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.99911140760507
Blobs2: 0.9207142539470758
Moons1: 0.6201236379808761
Moons2: 0.6240836112964199
Cirles1: 0.4996744496330887
Cirles2: 0.4996806760062264
```

Answer: The decreasing order of rand-index score for the datasets in the order Blobs1,Blobs2,Moons2,Moons1,Circles1,Circles2.

Question 1e: Are the rankings in (c) consistent with your observations in (d)? If not, explain the reason why your rankings were inconsistent.

Answer: Yes the rankings are in question c were consistent with the observations in question d.

2. Agglomerative Clustering - Single Link

Question 2a: Without running Single-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Single-link agglomerative clustering is expected to work well. Support your answer by explaining your rationale.

Answer: Single link clustering is expected to run well on the datasets Circles1_X,Moons1_X,Blobs1_X because in the single link the distance between the clusters (which was computed based on the distance between two points)is defined as the minimum distance between a point in one cluster and a point in another cluster.

In Blobs1_x each point is closer to all of the points in its cluster than to any point in another cluster, the data is spread in such a way that when applying the single link algorithm the clusters formed identical to the true cluster lables,when the singleton clusters were getting merged the adjacent points will get merged as the criteria for merging is minimum distance.so almost all three clusters were identified correctly.

In Moons1_x,circles1_X the points were oriented in such a way that there is no breakage or noise in between the points and points were aligned at a constant distance ,start with all points as singleton clusters and add links between points one at a time, shortest links first, then these single links combine the points into clusters and here the distance between the points in a particular cluster is less than the distance between any points in two clusters.

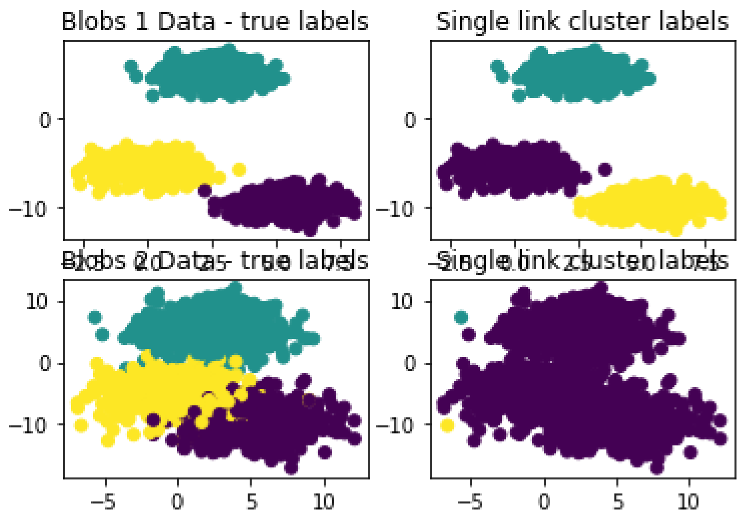
Question 2b: Without running Single-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Single-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

Answer: In Moons2_x,circles2_X ,Blobs2_X will start with all points as singleton clusters and add links between points one at a time, shortest links first, then these single links combine the points into clusters and after a while (few merges) the distance between the points in a particular cluster would become large when compared with the distance between the points in different clusters,so when merging instead of including the points those belonging to same cluster ,form a link with the points of the other cluster and then iterates.

Question 2c: Run Single-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Single-link agglomerative algorithm performance. Describe your rationale for your ranking.

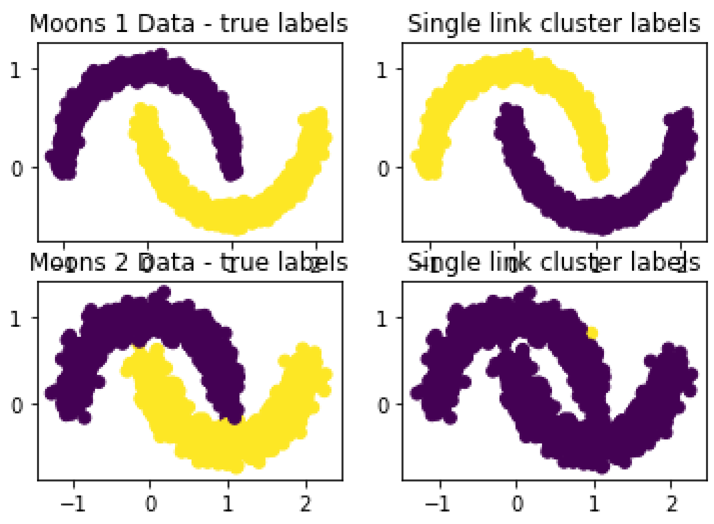
In [17]:

```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clusters)
y1_pred = single_linkage.fit_predict(Blobs1_X)
y2_pred = single_linkage.fit_predict(Blobs2_X)
plt.subplot(2,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('Single link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('Single link cluster labels')
plt.show()
```



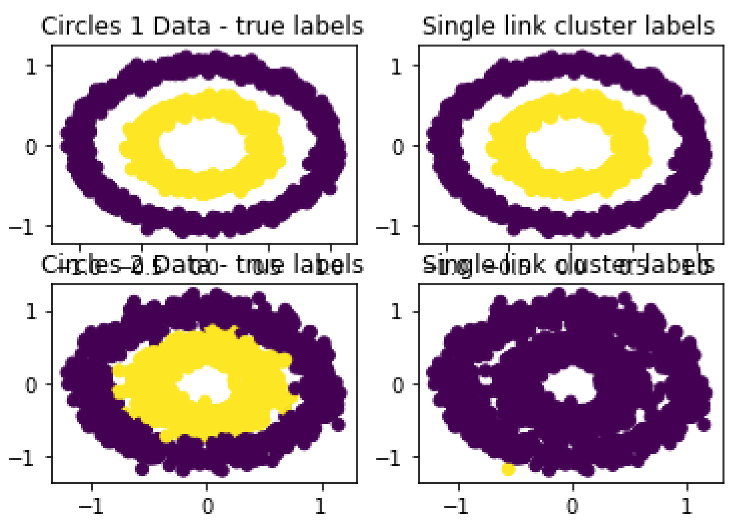
In [18]:

```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clusters)
y3_pred = single_linkage.fit_predict(Moons1_X)
y4_pred = single_linkage.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('Single link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('Single link cluster labels')
plt.show()
```



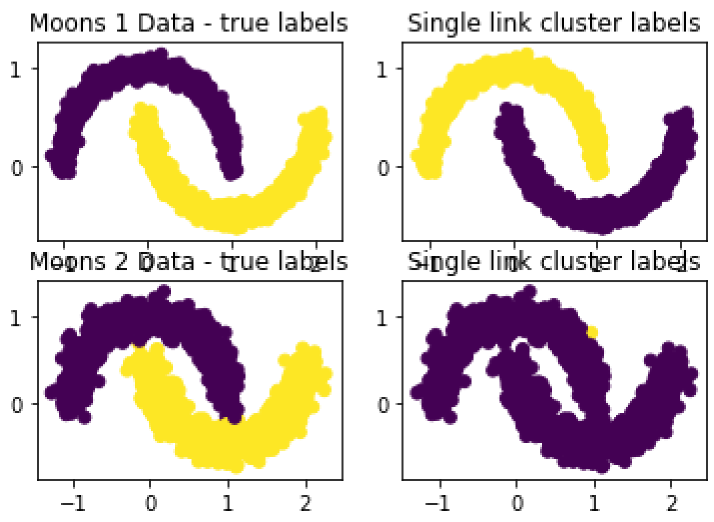
In [19]:

```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clusters)
y5_pred = single_linkage.fit_predict(Circles1_X)
y6_pred = single_linkage.fit_predict(Circles2_X)
plt.subplot(2,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title('Single link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('Single link cluster labels')
plt.show()
```



In [20]:

```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clusters)
y5_pred = single_linkage.fit_predict(Moons1_X)
y6_pred = single_linkage.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y5_pred)
plt.title('Single link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y6_pred)
plt.title('Single link cluster labels')
plt.show()
```



Answer: Based on the visualization, the decreasing order of Single-link agglomerative algorithm performance on the datasets is: Circles1_X,Moons1_X,Blobs1_x,Circles2_x,Moons2_X,Blobs2_x.

From the plots we can see that for Circles1_X,Moons1_X datasets the single link algorithm identified the clusters similar to true labels.

For Blobs1_X dataset,only two clusters were indetified even though there were 3 clusters becuase the distance between the points in bottom two clusters is minimum (at one instance) when compared with the distance between the points in a cluster so it merged the two clusters and able to detect almost 2 clusters out of 3 true clusters.

For Blobs1_X,Moons2_X,Circles2_X we could see from the graph that all the points were identified as a single cluster

Question 2d: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using Single-link agglomerative clustering algorithm. Rank the datasets in decreasing order of Rand-Index scores.

In [21]:

```
Rand_Blobs1=rand_index(y1_pred,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(y2_pred,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(y3_pred,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(y4_pred,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(y5_pred,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(y6_pred,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.99911140760507
Blobs2: 0.33377896375361354
Moons1: 1.0
Moons2: 0.49966733377807426
Cirles1: 1.0
Cirles2: 0.49966733377807426
```

Answer: The decreasing order of datasets based on the Rand Index scores were Moons1_X,Circles1_X,Blobs1_X,Circles2_X,Moons2_x,Blobs2_X.

Question 2e: Are the rankings in 2(c) consistent with your observations in 2(d)? If not, explain the reason why your rankings were inconsistent.

Answer: The rankings observed in question 2(c) were consistent with the observations in 2(d).

3. Agglomerative Clustering - Max Link

Question 3a: Without running Max-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Max-link agglomerative clustering is expected to work well. Support your answer by explaining your rationale.

Answer: The Complete link clustering is expected to work well on the datasets Blobs1_X,Blobs2_X because in the complete link ,the distance between the clusters defined as the maximum distance between the points in cluster one and cluster two.

when complete link algorithm applied on Blobs1_X,Blobs2_X datasets, single ton clusters will form when trying to merge the singleton clusters the two points clusters whose have minimum distance between the farther points as each point is closer to all of the points in its cluster than to any point in another cluster consecutively will get 3 three clusters which were exactly same as true clusters.

Question 3b: Without running Max-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Max-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

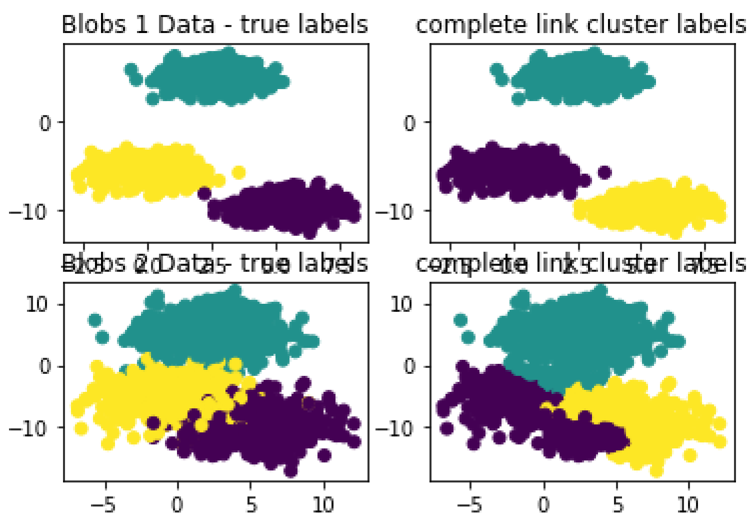
Answer: The Complete link clustering won't work well for the datasets Moons2_X,Moons1_X,Circles1_X,Circles2_X becuase when after half the of the points were merged the distance between the farthest point in the biggest cluster to the farthest point in smallest cluster is more when more between the distance between small clusters so in the each of the concentric circle two clusters will be formed and distance the between the farthest points in the concentric circles is less when compared between the farthest points with in concentric circle.so the clusters which are going to be form are not excatly as true clusters.

For Moons1_X and Moons2_X the farthest distance between the points in a (up or down faced moon) is more when compared between the fathest distance among the moons i.e inter farthest distance between the moons is less than the intra farthest distance among the moons so the complete link clustering form the clusters which were not same as true clusters.

Question 3c: Run Max-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Max-link agglomerative algorithm performance. Describe your rationale for your ranking.

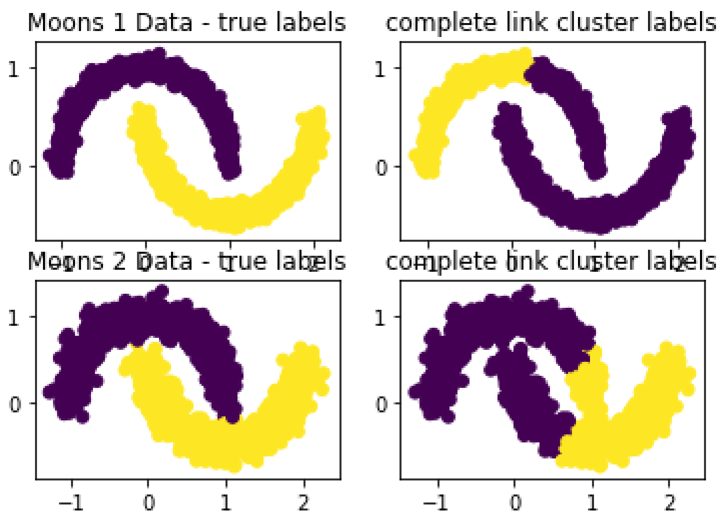
In [22]:

```
n_clusters = 3
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_clusters)
y1_pred = complete_linkage.fit_predict(Blobs1_X)
y2_pred = complete_linkage.fit_predict(Blobs2_X)
plt.subplot(2,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('complete link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('complete link cluster labels')
plt.show()
```



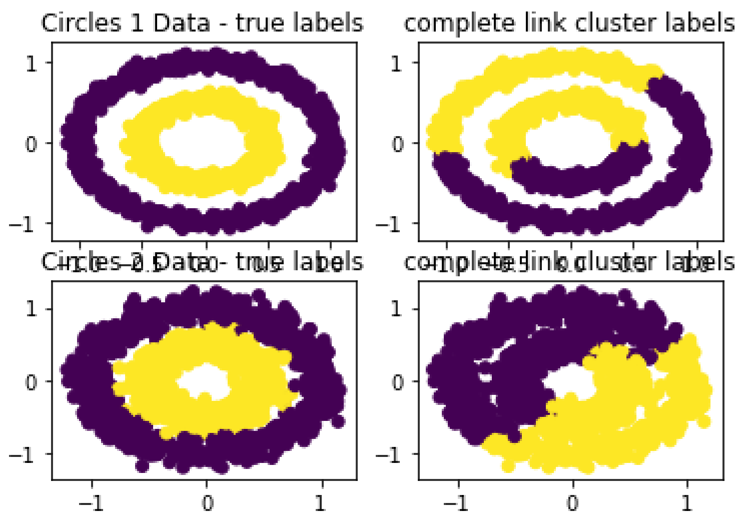
In [23]:

```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_clusters)
y3_pred = complete_linkage.fit_predict(Moons1_X)
y4_pred = complete_linkage.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('complete link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('complete link cluster labels')
plt.show()
```



In [24]:

```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_clusters)
y5_pred = complete_linkage.fit_predict(Circles1_X)
y6_pred = complete_linkage.fit_predict(Circles2_X)
plt.subplot(2,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title('complete link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('complete link cluster labels')
plt.show()
```



Answer: Based on the visualization the decreasing order of Max-link agglomerative algorithom performance on the datasets is Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X.

From the above we can infer that the clusters formed in Blobs1_X data is same as the true clusters lables and for Blobs2_X there were few variations when compared with true cluster labels particularly on the bottom two globular structures, less efficient when comapared with Blobs1_x data but more efficeint when compared with other data sets.

For Moons1_X,more than half of the points of a particular cluster are misclustered and for Moons2_X many points in total were misclustered (i.e some points from cluster1 and some points from cluster2)

For Circles1_X,Circles2_X almost half of the points in each true cluster were misclustered into a different cluster.

Question 3d: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using Max-link agglomerative clustering algorithm. Rank the datasets in decreasing order of Rand-Index scores.

In [25]:

```
Rand_Blobs1=rand_index(y1_pred,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(y2_pred,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(y3_pred,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(y4_pred,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(y5_pred,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(y6_pred,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.99911140760507
Blobs2: 0.7736544362908606
Moons1: 0.662605292417167
Moons2: 0.5965310206804536
Cirles1: 0.5218714698688014
Cirles2: 0.5000587058038692
```

Answer: The decreasing order of Rand-Index scores for the datssets is Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X when complete link algorithm is applied.

Question 3e: Are the rankings in 3(c) consistent with your observations in 3(d)? If not, explain the reason why your rankings were inconsistent.

Answer: Yes the rankings in 3c were consistent with 3d rand index values.

4. Agglomerative Clustering - Average Link

Question 4a: Without running Average-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Average-link agglomerative clustering is expected to work well. Support your answer by explaining your rationale.

Answer: In Average link agglomerative clustering , closest pair of clusters were formed based on the proximity the average pairwise distance between the points in the clusters.This algorithm works better for Blobs1_X because when calculating the clusters based on the average link and this method can handle the noise in the data .

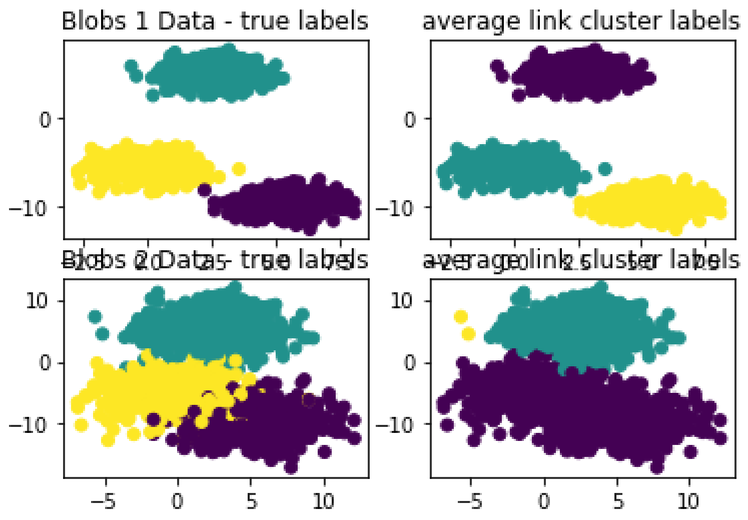
Question 4b: Without running Average-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Average-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

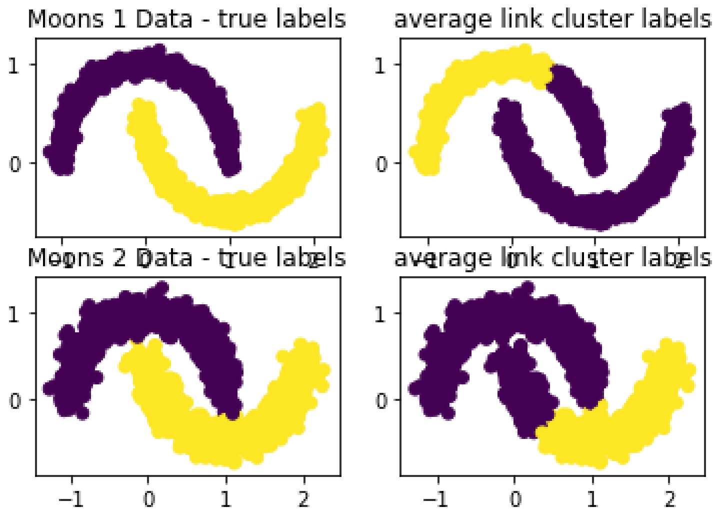
Answer: Average link clustering won't work well on the datasets Blobs2_X,Circles1_X,Circles2_X,Moons1_X,Moons2_X because it won't work well on curve data as every subset of vectors have a different cohesion.

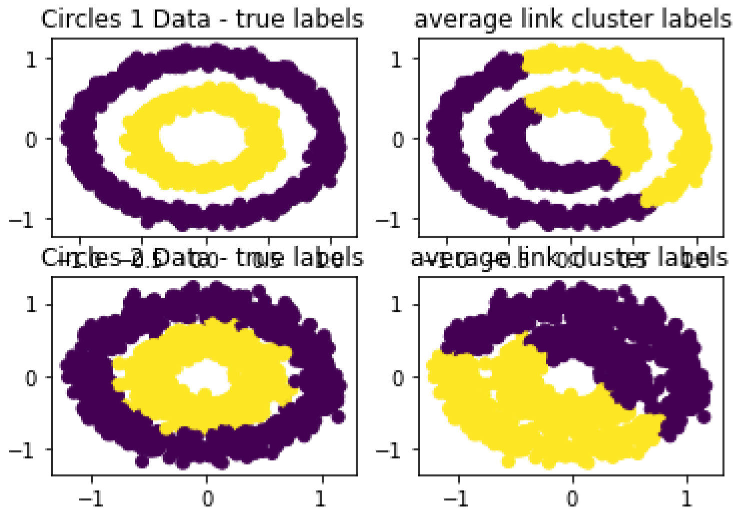
Question 4c: Run Average-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Average-link agglomerative algorithm performance. Describe your rationale for your ranking.

In [55]:

```
n_clusters = 3
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clusters)
y1_pred = average_linkage.fit_predict(Blobs1_X)
y2_pred = average_linkage.fit_predict(Blobs2_X)
plt.subplot(2,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('average link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('average link cluster labels')
plt.show()


n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clusters)
y3_pred = average_linkage.fit_predict(Moons1_X)
y4_pred = average_linkage.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('average link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('average link cluster labels')
plt.show()

n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clusters)
y5_pred = average_linkage.fit_predict(Circles1_X)
y6_pred = average_linkage.fit_predict(Circles2_X)
plt.subplot(2,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title(' average link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('average link cluster labels')
plt.show()
```


Blobs 1 Data - true labels
average link cluster labels
Blobs 2 Data - true labels
average link cluster labels


Moons 1 Data - true labels
average link cluster labels
Moons 2 Data - true labels
average link cluster labels


Circles 1 Data - true labels
average link cluster labels
Circles 2 Data - true labels
average link cluster labels

Answer: Based on the visualization the decreasing order of Max-link agglomerative algorithom performance on the datasets is Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X.

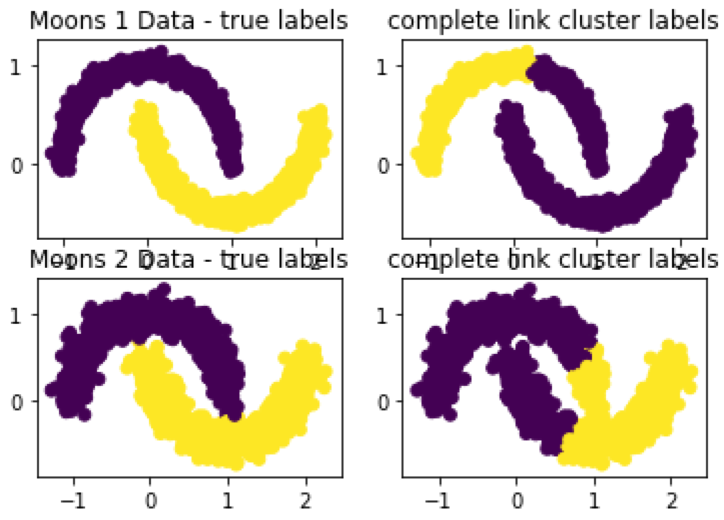
From the above we can infer that the clusters formed in Blobs1_X data is same as the true clusters lables and for Blobs2_X there were few variations when compared with true cluster labels particularly on the bottom two globular structures, less efficient when comapared with Blobs1_x data but more efficeint when compared with other data sets.

For Moons1_X,more than half of the points of a particular cluster are misclustered and for Moons2_X very few points were misclustered of a particular cluster and the clusters formed were more similar ot its ture clusters.

For Circles1_X,Circles2_X almost half of the points in each true cluster were misclustered into a different cluster.

In [56]:

```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_clusters)
y3_pred = complete_linkage.fit_predict(Moons1_X)
y4_pred = complete_linkage.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('complete link cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('complete link cluster labels')
plt.show()
```



Question 4d: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using Average-link agglomerative clustering algorithm. Rank the datasets in decreasing order of Rand-Index scores.

In [57]:

```
Rand_Blobs1=rand_index(y1_pred,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(y2_pred,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(y3_pred,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(y4_pred,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(y5_pred,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(y6_pred,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.99911140760507
Blobs2: 0.7636575494774294
Moons1: 0.662605292417167
Moons2: 0.5965310206804536
Cirles1: 0.500414498554592
Cirles2: 0.5050780520346898
```

Answer:

The decreasing order of Rand-Index scores for the datssets is Blobs1_X,Blobs2_X,Moons2_X,Moons1_X,Circles1_X,Circles2_X when Average link algorithm is applied.

Question 4e: Are the rankings in 4(c) consistent with your observations in 4(d)? If not, explain the reason why your rankings were inconsistent.

Answer: Yes the rankings were in 4(c) are consistent with the rankings in 4d.

5. Density Based Clustering: DBSCAN

Question 5a: Without running DBSCAN clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where DBSCAN clustering is expected to work well. Support your answer by explaining your rationale.

Answer: The DBScan clustering is expected to work well for the datasets Blobs1_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X as it uses the local density of points to determine the clusters ,rather than using only the distance between points.

Here for the datasets Blobs1_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X the density of the points with in the cluster points is more and have few breakges(when we have breakge in data i.e desnity less than minsup ,we identify the points near to the breakage as either border or noise points so there in no possiblity of extending DFS walk) in densities between the clusters so that it will identify those points were not belong to that cluster.

Question 5b: Without running DBSCAN clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where DBSCAN clustering is expected to NOT work well. Support your answer by explaining your rationale.

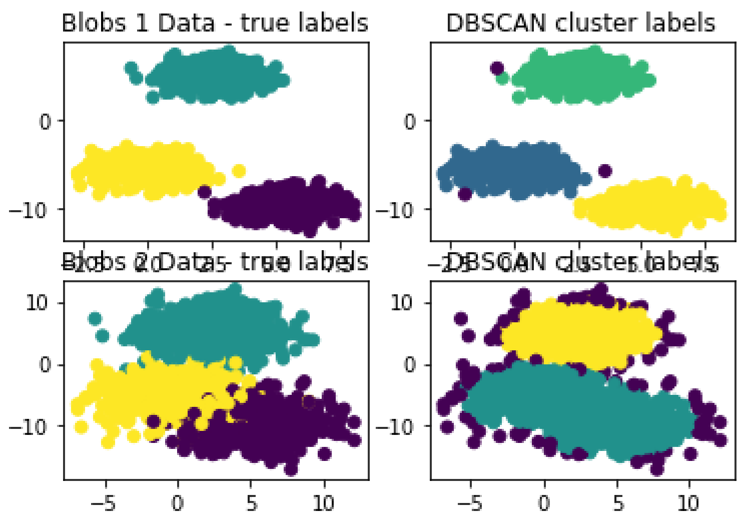
Answer:

The DBSCAN clustering is not expected to work on the dataset Blobs2_X because the density of the points with in the cluster points is more and goes on increasing and there was no breakages in the data i.e density is always more than minsup and it will consider all points as core points and DFS continue to extend to form a big clusters as the denser clusters will merge.so there might be possibility of getting merge of all the clusters into one.

Question 5c: Run DBSCAN clustering algorithm on all the datasets (except Rand). **Choose eps and min_samples parameters to make sure that DBSCAN finds the same number of clusters as in the ground truth ('Data_y').** Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of DBSCAN clustering algorithm performance. Describe your rationale for your ranking.

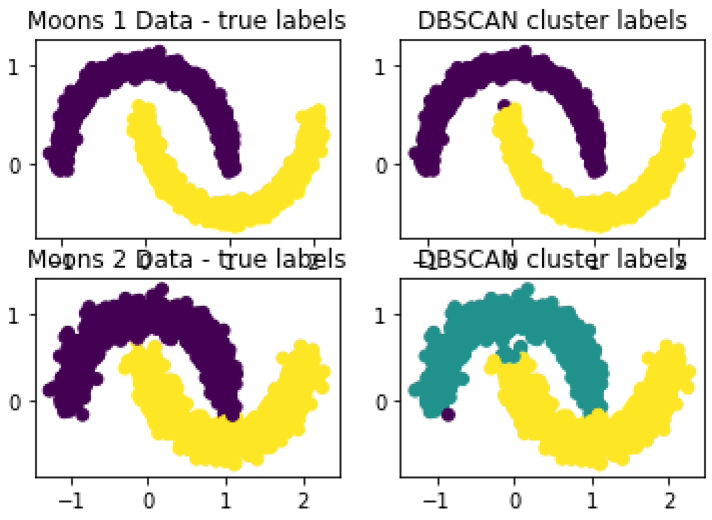
In [58]:

```
dbscan= DBSCAN(eps=1, min_samples=10)
dbscan_B2= DBSCAN(eps=2.5, min_samples=120)
y1_pred = dbscan.fit_predict(Blobs1_X)
y2_pred = dbscan_B2.fit_predict(Blobs2_X)
plt.subplot(2,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('DBSCAN cluster labels')
plt.subplot(2,2,3)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('DBSCAN cluster labels')
plt.show()
```



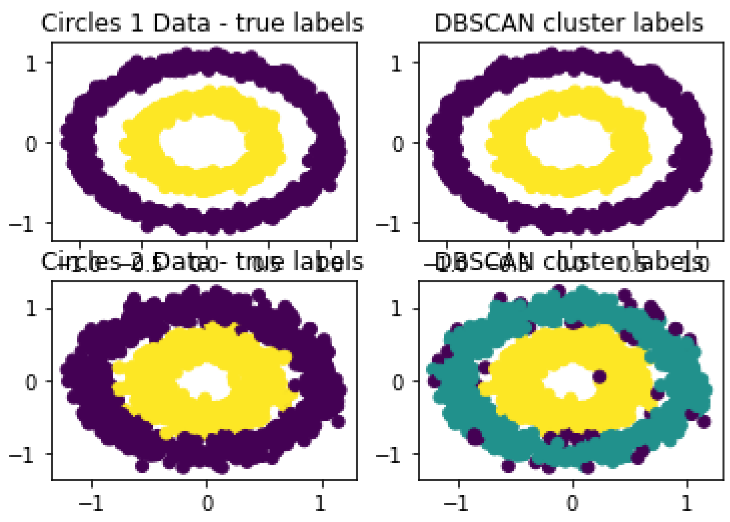
In [59]:

```
dbscan= DBSCAN(eps=0.3, min_samples=100)#0.2,10
y3_pred = dbscan.fit_predict(Moons1_X)
y4_pred = dbscan.fit_predict(Moons2_X)
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('DBSCAN cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('DBSCAN cluster labels')
plt.show()
```



In [60]:

```
dbscan= DBSCAN(eps=0.1, min_samples=8)#0.1 8 #0.3,150
dbscan_c2= DBSCAN(eps=0.1, min_samples=8)
y5_pred = dbscan.fit_predict(Circles1_X)
y6_pred = dbscan_c2.fit_predict(Circles2_X)
plt.subplot(2,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title('DBSCAN cluster labels')
plt.subplot(2,2,3)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('DBSCAN cluster labels')
plt.show()
```



Answer: Based on the visualization the decreasing order of ranks when DBSCAN applied on the datasets is Circles1_X,Blobs1_X,Moons1_X,Moons2_X,Circles2_x,Blobs2_X .

From the above plots we can infer that DBSCAN algorithm able to classify the clusters sucessfully for Moons1_X,Circle1_X,Blobs1_X,Moons2_X,circles2_X,Blobs2_X when we choose the correct eps and min_samples parameters as the points in these datasets were closely densed.

Question 5d: For each of the datasets, how many noise points did the DBSCAN algorithm find? Which three datasets had the least number of noise points? Explain the reason(s) why these datasets had least noise points?

In [61]:

```
print('Noise points in Blobs1_X :' +str (np.sum(y1_pred==-1)))
print('Noise points in Blobs2_X :' +str (np.sum(y2_pred==-1)))
print('Noise points in Moons1_X:' +str (np.sum(y3_pred==-1)))
print('Noise points in Moons2_X:' +str (np.sum(y4_pred==-1)))
print('Noise points in Circles1_X:' +str (np.sum(y5_pred==-1)))
print('Noise points in Circles2_X:' +str (np.sum(y6_pred==-1)))
```

```
Noise points in Blobs1_X :3
Noise points in Blobs2_X :118
Noise points in Moons1_X:0
Noise points in Moons2_X:1
Noise points in Circles1_X:0
Noise points in Circles2_X:48
```

Answer: Circles1_X,Moons1_X,Moons2_X are the datasets with least number of noise points because in these datasets the points were close to each other and all the points were with in the radius of eps value with respect to other points.

Question 5e: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using DBSCAN clustering algorithm. Rank the datasets in decreasing order of Rand-Index scores.

In [62]:

```
m=np.where(y1_pred==-1, 100, y1_pred)
n=np.where(y2_pred==-1, 100, y2_pred)
o=np.where(y3_pred==-1, 100, y3_pred)
p=np.where(y4_pred==-1, 100, y4_pred)
q=np.where(y5_pred==-1, 100, y5_pred)
r=np.where(y6_pred==-1, 100, y6_pred)
```

In [63]:

```
m=np.where(y1_pred==-1, 100, y1_pred)
n=np.where(y2_pred==-1, 100, y2_pred)
o=np.where(y3_pred==-1, 100, y3_pred)
p=np.where(y4_pred==-1, 100, y4_pred)
q=np.where(y5_pred==-1, 100, y5_pred)
r=np.where(y6_pred==-1, 100, y6_pred)
Rand_Blobs1=rand_index(m,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(n,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(o,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(p,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(q,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(r,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.997781632199244
Blobs2: 0.7505643762508339
Moons1: 0.9986666666666667
Moons2: 0.9847747387146987
Cirles1: 1.0
Cirles2: 0.9590802757393818
```

Answer: The decreasing order of rand index scores when applied on the output of DBscan algorithm for the datasets is Circles1_X,Blobs1_X,Moons1_X,Moons2_X,Circles2_x,Blobs2_X

Question 5f: Are the rankings in 5(c) consistent with your observations in 5(e)? If not, explain the reason why your rankings were inconsistent.

Answer: yes the rankings in 5(c) are consistent with 5(e).

6. Spectral Clustering

Question 6a: Without running Spectral clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Spectral clustering is expected to work well. Support your answer by explaining your rationale.

Answer: Spectral clustering is a flexible approach for finding clusters when the data doesn’t meet the requirements of other common algorithms. The Spectral clustering alogorithms works efficiently for all the datasets because we are projecting the points into lower space and then will apply corresponding clustering algorithm (like K means for globular data,DBSCAN for bean shaped/ halfmoons) which would be suited for the data.

Here in the practice session Spectral clustering algorithm works better on the datasets Blobs1_X,Blobs2_X,Circles1_X,Moons1_X.

For Blobs1_X,Blobs2_X after finding the affinity matrix and degree matrix will find the eigen vector,eigen values and then based on the number of clusters will select those many max/min eigenvectors based on the cut and apply K means algorithm on the data. For Blobs1_X,Blobs2_X the internal affinity with in the clusters is high and also the external affinity with in clusters is less.so spectral clustering works better here.

For Circles1_X,Moons1_X, after finding the affinity matrix and degree matrix will find the eigen vector,eigen values and then based on the number of clusters will select those many max/min eigenvectors based on the cut and apply DB scan algorithm on the data with suitable parameters.For Circles1_X,Moons1_X the internal affinity with in the clusters is high and also the external affinity with in clusters is less.so spectral clustering works better here.

Question 6b: Without running Spectral clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Spectral clustering is expected to NOT work well. Support your answer by explaining your rationale.

Answer: Spectral clustering Won't work well for the datasets Circles2_X,Moons2_X because the internal affinity is high but the external affinity between the clusters is less.

So spectral clustering won't work well on the datasets Circles2_X,Moons2_X.

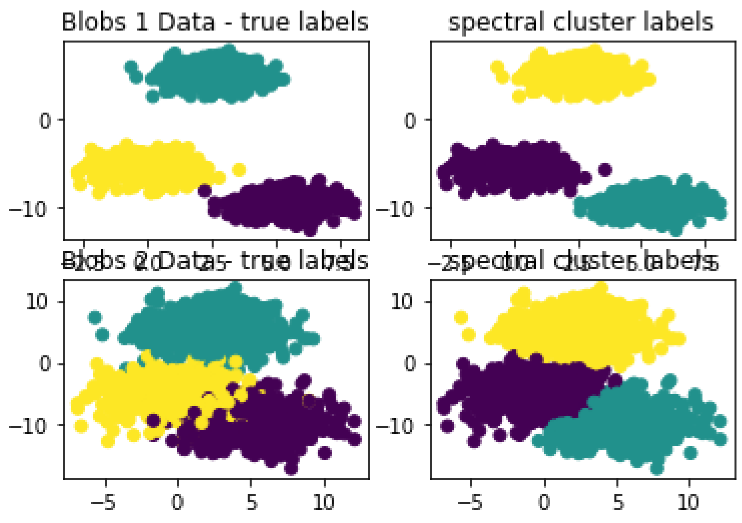
Question 6c: Run Spectral clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Spectral clustering algorithm performance. Describe your rationale for your ranking.

In [64]:

```
def aff(DS):
    k=len(DS)
    A=np.zeros((k,k))
    for i in range(len(A)):
        for j in range(len(A)):
            p=DS[i]
            q=DS[j]
            n=np.dot(p,q)
            A[i][j]=n
    x=[]
    v=0
    for i in range(len(A)):
        for j in range(len(A)):
            v=A[i][j]+v
        x.append(v)
    D=np.diagflat(x)
    return A,D
```

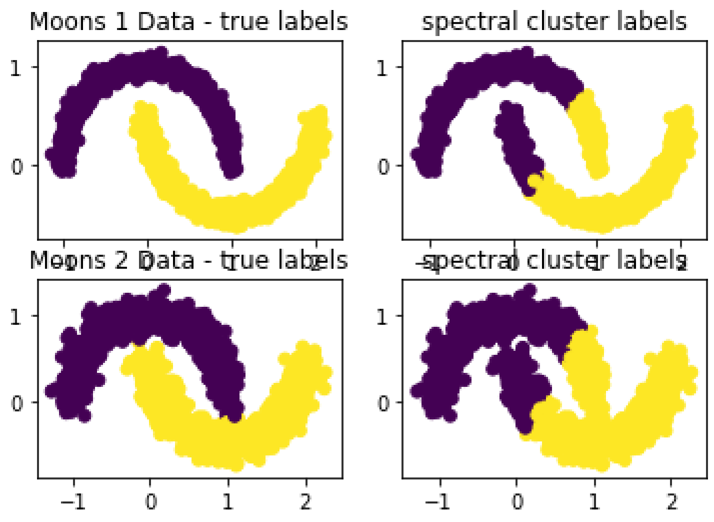
In [65]:

```
n_clusters=3
clustering1 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Blobs1_X)
y1_pred = clustering1.labels_
clustering2 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Blobs2_X)
y2_pred = clustering2.labels_
plt.subplot(2,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y1_pred)
plt.title('spectral cluster labels')
plt.subplot(2,2,3)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y2_pred)
plt.title('spectral cluster labels')
plt.show()
```



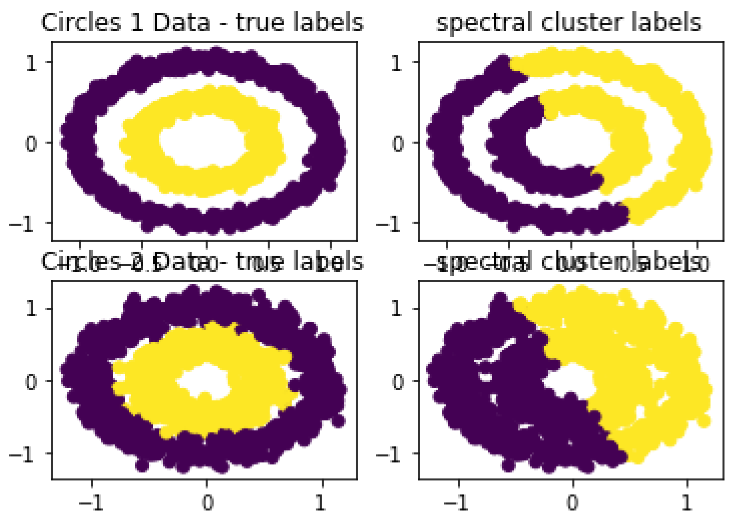
In [66]:

```
n_clusters=2
clustering3 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Moons1_X)
y3_pred = clustering3.labels_
clustering4 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Moons2_X)
y4_pred = clustering4.labels_
plt.subplot(2,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y3_pred)
plt.title('spectral cluster labels')
plt.subplot(2,2,3)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y4_pred)
plt.title('spectral cluster labels')
plt.show()
```



In [67]:

```
clustering5 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Circles1_X)
y5_pred = clustering5.labels_
clustering6 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Circles2_X)
y6_pred = clustering6.labels_
plt.subplot(2,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles 1 Data - true labels')
plt.subplot(2,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y5_pred)
plt.title('spectral cluster labels')
plt.subplot(2,2,3)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles 2 Data - true labels')
plt.subplot(2,2,4)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y6_pred)
plt.title('spectral cluster labels')
plt.show()
```



Answer: By default the Spectral clustering applies the k means algorithm on the eigen vectors and the decreasing order of spectral clustering algorithm performance on the datasets is Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X.

using K means on Blobs1_X,Blobs2_X clusters were formed exactly as of the true clusters and K means algorithm suited for globular data.

For Moons1_X,Moons2_X more than 50% of the points were clustered correctly in each cluster ,the distance between the extreme points of a particular moon is more when the compared with extreme points distance between the moons as it is distance based algorithm it misclusted the extreme points .

For Circles1_x,Circles2_X only few points were clustered correctly when compared with other datasets .

Question 6d: For each of the datasets, compute Rand-Index value between the true labels and cluster memberships computed using Spectral clustering algorithm. Rank the datasets in decreasing order of Rand-Index scores.

In [68]:

```
Rand_Blobs1=rand_index(y1_pred,Blobs1_y)
print('Blobs1: ' +str(Rand_Blobs1))
Rand_Blobs2=rand_index(y2_pred,Blobs2_y)
print('Blobs2: ' +str(Rand_Blobs2))
Rand_Moons1=rand_index(y3_pred,Moons1_y)
print('Moons1: ' +str(Rand_Moons1))
Rand_Moons2=rand_index(y4_pred,Moons2_y)
print('Moons2: ' +str(Rand_Moons2))
Rand_circles1=rand_index(y5_pred,Circles1_y)
print('Cirles1: ' +str(Rand_circles1))
Rand_circles2=rand_index(y6_pred,Circles2_y)
print('Cirles2: ' +str(Rand_circles2))
```

```
Blobs1: 0.99911140760507
Blobs2: 0.9191825661552145
Moons1: 0.6441263064265066
Moons2: 0.6448441183010896
Cirles1: 0.4996806760062264
Cirles2: 0.499710028908161
```

Answer: The decreasing order of randindex values after peroforming spectral clustering algorithm on the data is Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X.

Question 6e: Are the rankings in 6(c) consistent with your observations in 6(d)? If not, explain the reason why your rankings were inconsistent.

Answer: Yes the rankings in 6(c) were consistent with 6(d).

7. Clustering Tendency

Question 7a: Without using any metrics, for all the datasets (INCLUDING **Rand**) provided in the practice session, list the datasets that exhibit good clustering tendency. Support your answer by explaining your rationale.

Answer: Blobs1_X,Blobs2_X,Moons1_X,Moons2_X,Circles1_X,Circles2_X datasets have the tendency to exhibit good clustering ,when we visualize the data we could see the meaningful clusters in the data.

If the data is in high dimensions will do the dimensionality reduction and visualize the data in lower dimesnions and conculde whether we see the meaningful clusters in the data.

Question 7b: Without using any metrics, for all the datasets (INCLUDING Rand) provided in the practice session, list the datasets that do NOT exhibit good clustering tendency. Support your answer by explaining your rationale.

Answer: Rand_X dataset don't exhibit the good clustering tendency because the randomly generated data set and doesn’t contain any meaningful clusters.

Question 7c: Compute Hopkins Statistic statistic for all the datasets and rank them based on decreasing order of this metric.

In [69]:

```
print('Hopkins statistics for Blobs1_X : ' +str(hopkins(Blobs1_X)))
print('Hopkins statistics for Blobs2_X : ' +str(hopkins(Blobs2_X)))
print('Hopkins statistics for Moons1_X : ' +str(hopkins(Moons1_X)))
print('Hopkins statistics for Moons2_X : ' +str(hopkins(Moons2_X)))
print('Hopkins statistics for Circles1_X : ' +str(hopkins(Circles1_X)))
print('Hopkins statistics for Circles2_X : ' +str(hopkins(Circles2_X)))
print('Hopkins statistics for RAND_X : ' +str(hopkins(Rand_X)))
```

```
Hopkins statistics for Blobs1_X : 0.93418629777965
Hopkins statistics for Blobs2_X : 0.8501976924522093
Hopkins statistics for Moons1_X : 0.921879032024323
Hopkins statistics for Moons2_X : 0.873131985542685
Hopkins statistics for Circles1_X : 0.8580839866509072
Hopkins statistics for Circles2_X : 0.7633459162162567
Hopkins statistics for RAND_X : 0.602562588271576
```

Answer: The decreasing order of the Hopkins statistic on the datasets is Blobs1_X,Moons1_X ,Moons2_X ,Blobs2_x,Circles1_X,Circles2_X,Rand_X

Question 7d: Are your answers for 7(a) and 7(b) consistent with that of (c)? If not, explain the reason for this inconsistency.

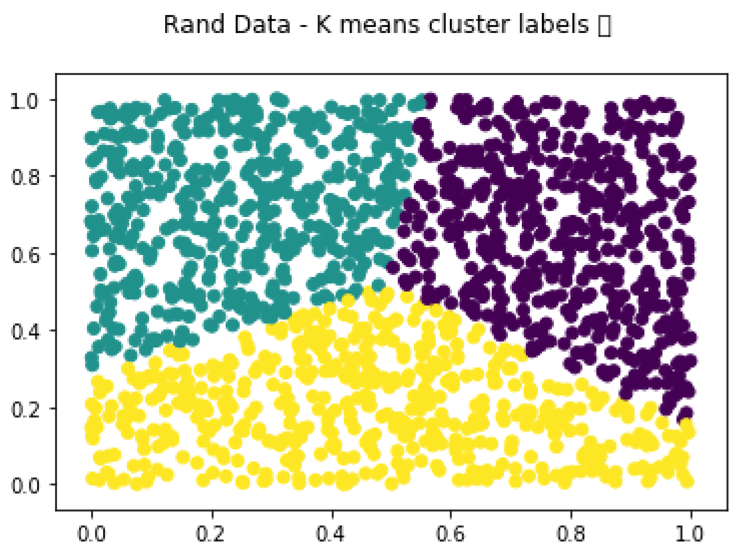
Answer: Yes the answers in 7(a) are consistent with 7(b)

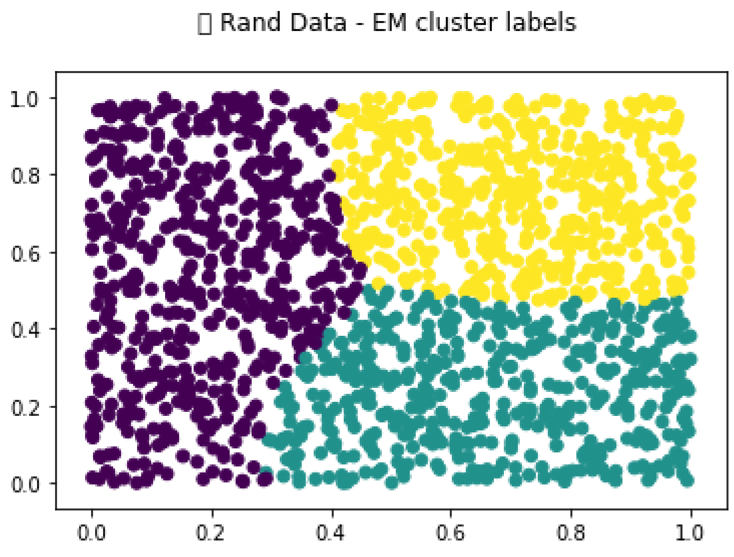
Question 7e: Run all the above clustering algorithms (KMeans, GMM, Agglomerative (single, max, average), DBSCAN, Spectral), using n_clusters = 3, on Rand dataset and visualize the clusters. Explain the reason for the shapes of clusters dervied using each clustering approach.

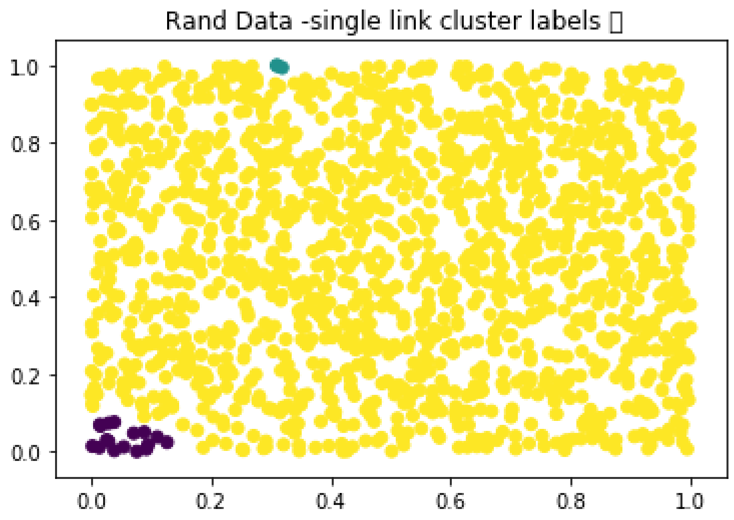
In [70]:

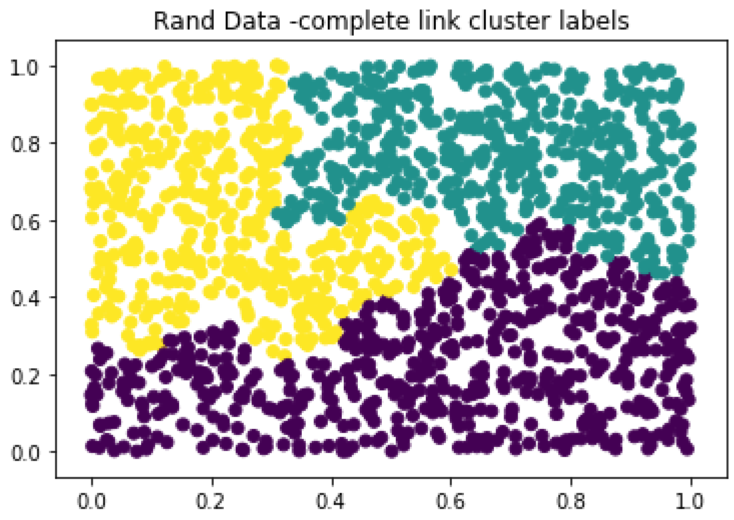
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y1_pred=kmeans.fit_predict(Rand_X)
gmm = GaussianMixture(n_components=n_clusters, covariance_type='full')
y2_pred = gmm.fit_predict(Rand_X)
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clusters)
y3_pred = single_linkage.fit_predict(Rand_X)
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_clusters)
y4_pred = complete_linkage.fit_predict(Rand_X)
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clusters)
y5_pred = average_linkage.fit_predict(Rand_X)
dbscan = DBSCAN(eps=0.005, min_samples=15)
y6_pred = dbscan.fit_predict(Rand_X)
clustering1 = SpectralClustering(n_clusters=n_clusters, assign_labels="discretize", ran
dom_state=0).fit(Rand_X)
y7_pred = clustering1.labels_
#plt.subplot(3,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y1_pred)
plt.title(f'Rand Data - K means cluster labels \t \n ')
plt.show()
print("\t"  )
#plt.subplot(3,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y2_pred)
plt.title(f'\t Rand Data - EM cluster labels \n')
plt.show()
print("\n")
#plt.subplot(3,2,3)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y3_pred)
plt.title(f' Rand Data -single link cluster labels \t')
plt.show()
#plt.subplot(3,2,4)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y4_pred)
plt.title('Rand Data -complete link cluster labels')
plt.show()
#plt.subplot(3,2,5)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y5_pred)
plt.title('Rand Data -average link cluster labels')
plt.show()
#
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y6_pred)
plt.title('Rand Data -DBSCAN cluster labels')
plt.show()
#plt.subplot(3,2,6)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y7_pred)
plt.title('Rand Data -Spectral cluster labels')
plt.show()
```

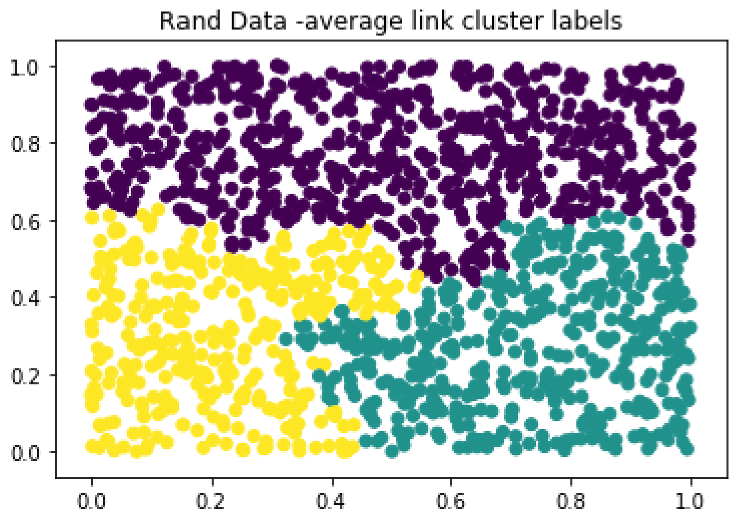
```
/users/PES0801/dogipakr/.local/lib/python3.6/site-packages/matplotlib/back
ends/backend_agg.py:211: RuntimeWarning: Glyph 9 missing from current fon
t.
  font.set_text(s, 0.0, flags=flags)
/users/PES0801/dogipakr/.local/lib/python3.6/site-packages/matplotlib/back
ends/backend_agg.py:180: RuntimeWarning: Glyph 9 missing from current fon
t.
  font.set_text(s, 0, flags=flags)
```

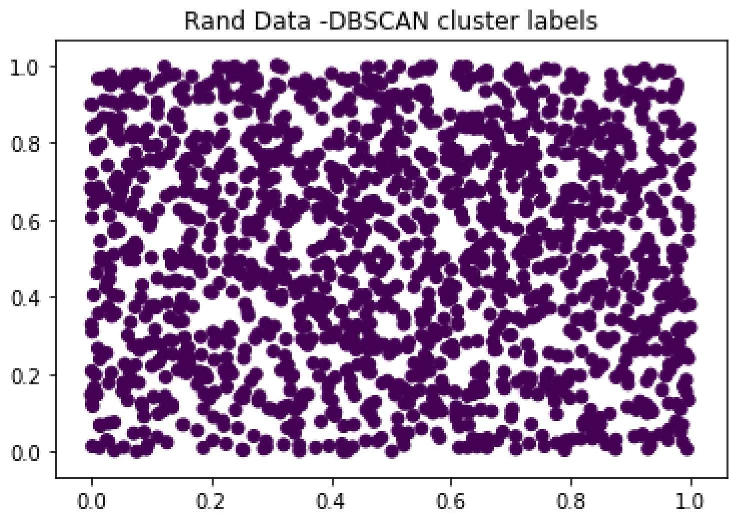


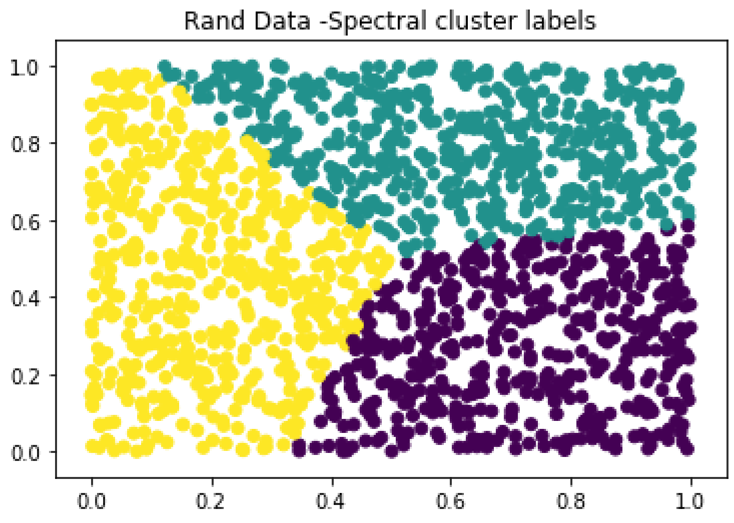



Rand Data -single link cluster labels
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Rand Data -complete link cluster labels
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Rand Data -average link cluster labels
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Rand Data -DBSCAN cluster labels
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



Answer: For K means clustering graph as K value is 3 the graph shows three clusters with minimum sse scores. Here it will select three means randomly and will assign the points to a centroid whose is distance is minimum ,this process is iterated untill the mean values were converged,as a result 3 clusters were formed and the point at (0.5,0.5)will belongs to the blue cluster instead of yellow or cyan is because the point is at a distance minimum to the blue cluster centriod.

For EM algorithm once the model parameters were intialized will try to maximize the likelihood probality after every iterations and will assign to class with maximum likelihood .so the points blue cluster has more likelihood probablity when calculated with the model (newly calculated mean and variance).

For Single link algorithm ,it will form clusters based on the minimum distance between the closet points of the clusters.here the distance is less ,so it formed considered an entire data as one cluster.

For complete linke ,it will form the closet pair of clusters based on the distance between the farthest points of the clusters.why here the clusteres were not merged because suppose for a point in the cluster at 0.42,0.37 why it didn't mapped to blue color is the distance between the farthest point to that point is more when compared to the farthest point to that point in yellow cluster.so 3 clusters were formed

For avergae clustering technique the avergae distance is calculated at each time and will megre a cluster with minimum avergae distance between the cluster.

For DBSCAN algorithm as the points were spread were closely and the density among the points is very high even a distance of 0.05 we have 5 points so it considered everything as one cluster , only one cluster is formed.

For spectral clsutering ,the affinity between the points is very high even after projecting to lowdimension also we get the clusters with similar results as K means,hence 3 clusters were formed.

8. Real-world dataset

We will use the same breast cancer dataset we used for Classificatione exercise here.

In [71]:

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The features are:

In [72]:

```
cancer.feature_names
```

Out[72]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

Class labels are:

In [73]:

```
cancer.target_names
```

Out[73]:

```
array(['malignant', 'benign'], dtype='<U9')
```

Create dataset for classification

In [74]:

```
Cancer_X = cancer.data
Cancer_y = cancer.target
```

Size of Cancer_X and Cancer_y

In [75]:

```
Cancer_X.shape
```

Out[75]:

```
(569, 30)
```

In [76]:

```
Cancer_y.shape
```

Out[76]:

```
(569,)
```

Question 8a: Compute SSE for k = range(2,40), i.e, for k=2,3,4,...,40

In [77]:

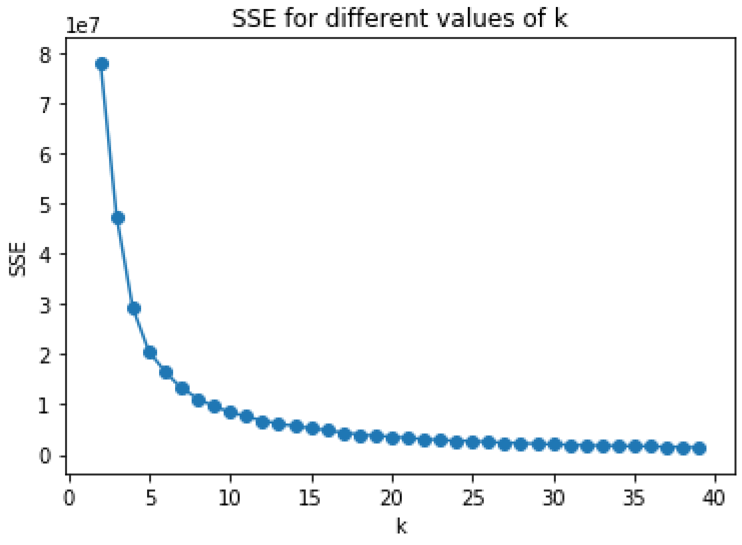
```
score = np.zeros(40);
for i in range(2,40):
 kmeans = KMeans(n_clusters=i, random_state=random_state); #Initializing KMeans
 kmeans.fit_predict(Cancer_X) #Clustering using KMeans
 score[i] = -kmeans.score(Cancer_X) #Computing SSE
 print("SSE for k=",i,":", round(score[i],2)) #Printing SSE
```

```
SSE for k= 2 : 77943099.88
SSE for k= 3 : 47285926.9
SSE for k= 4 : 29226541.65
SSE for k= 5 : 20539877.62
SSE for k= 6 : 16558716.7
SSE for k= 7 : 13249736.07
SSE for k= 8 : 11183535.78
SSE for k= 9 : 9609383.58
SSE for k= 10 : 8487166.05
SSE for k= 11 : 7613587.21
SSE for k= 12 : 6784588.86
SSE for k= 13 : 6157087.42
SSE for k= 14 : 5708365.13
SSE for k= 15 : 5286031.4
SSE for k= 16 : 4848940.46
SSE for k= 17 : 4398276.58
SSE for k= 18 : 4009831.04
SSE for k= 19 : 3738118.1
SSE for k= 20 : 3578729.29
SSE for k= 21 : 3312041.59
SSE for k= 22 : 3102392.22
SSE for k= 23 : 2894387.69
SSE for k= 24 : 2768624.68
SSE for k= 25 : 2685795.48
SSE for k= 26 : 2514580.5
SSE for k= 27 : 2362959.95
SSE for k= 28 : 2257591.62
SSE for k= 29 : 2148955.49
SSE for k= 30 : 2036764.0
SSE for k= 31 : 1969448.35
SSE for k= 32 : 1833170.63
SSE for k= 33 : 1791369.0
SSE for k= 34 : 1722589.76
SSE for k= 35 : 1677340.13
SSE for k= 36 : 1656114.54
SSE for k= 37 : 1528956.63
SSE for k= 38 : 1496563.45
SSE for k= 39 : 1417439.02
```

Question 8b: Plot SSE values for k = range(2,40), i.e, for k=2,3,4,...,40

In [78]:

```
plt.plot(range(2,40),score[2:40])
plt.scatter(range(2,40),score[2:40])
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('SSE for different values of k')
plt.show()
```



Question 8c: Using this plot, determine the 'k' that you will use to do K-Means clustering.

Answer: K value would be 7 from the plot as there is knee jerk in SSE,k graph at k value 7.so K value would be taken as 7.

Question 8d: Using the 'k' you chose in (c), compute k-Means clustering.

In [79]:

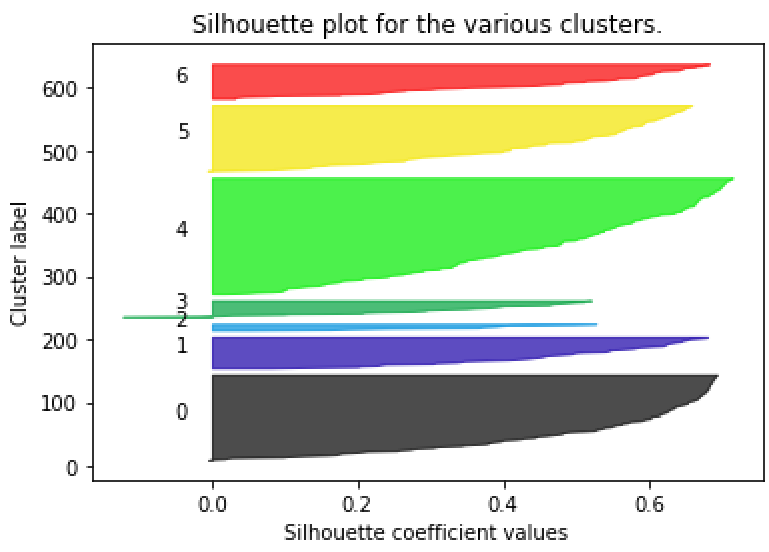
```
n_clusters = 7
random_state=10
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y1_pred = kmeans.fit_predict(Cancer_X)
print(np.unique(y1_pred))
```

```
[0 1 2 3 4 5 6]
```

Question 8e: Plot the silhouette values for points in each cluster (using the silhouette() function provided in the practice notebook). .

In [80]:

```
silhouette(Cancer_X,y1_pred)
```



Question 8f: Comment on the quality of the clusters discovered using k-Means. Which of the clusters would you treat as good clusters and which clusters do you treat as not-so-good clusters?

Answer:

Cluster 4 , cluster 0,cluster6, are good clusters because they good cohesion and seperation and clusters2,clusters3,clusters1 are not good clusters because the cohesion and seperation for the clusters is less.

Question 8g: Compute the Rand Index of the k-means clusters with respect to the true labels. Comment on the quality of the clustering based on the Rand-Index score.

In [81]:

```
k=rand_index(y1_pred,Cancer_y)
print(k)
```

```
0.6167231862174807
```

Answer: As Rand Index is 0.6167231862174807 ,the K means algorithm correctly clustered only 61% of the points.

Question 8h: To use DBSCAN to find clusters in this data, one needs to determine eps and min_samples. To do this, consider the range of values eps = 50, 100, 150, 200, 250, 300, 400, 500 and min_samples = 10, 15, 20, 25, 30.

For these range of eps and min_samples values, compute an 8x5 matrix (with rows as eps values and cols as min_samples) to show the number of clusters obtained at each of these parameters. Visualize this matrix using imshow() in matplotlib.

Hint: To compute the number of clusters, you may use:

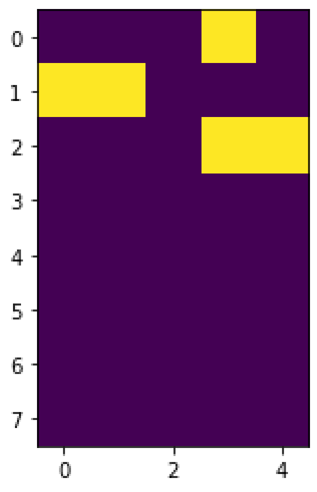
y_pred = dbscan.fit_predict(Cancer_X)

max(y_pred)+1

In [82]:

```
#A=np.zeros((k,k))
eps = [50, 100, 150, 200, 250, 300, 400, 500] #10 &100
min_samples =[10, 15, 20, 25, 30]
clusters=np.zeros((8,5))
for i in range(8):
    for j in range(5):
        dbscan= DBSCAN(eps=eps[i], min_samples=min_samples[j])
        y_pred = dbscan.fit_predict(Cancer_X)
        clusters[i][j]=max(y_pred)+1
print(clusters)
plt.imshow(clusters)
plt.show()
```

```
[[1. 1. 1. 2. 1.]
 [2. 2. 1. 1. 1.]
 [1. 1. 1. 2. 2.]
 [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.]]
```



Question 8i: For these range of eps and min_samples values, compute an 8x5 matrix (with rows as eps values and cols as min_samples) to show the number of noise points obtained at each of these parameters. Visualize this matrix using imshow() in matplotlib.

Hint: To compute the number of noise points, you may use:

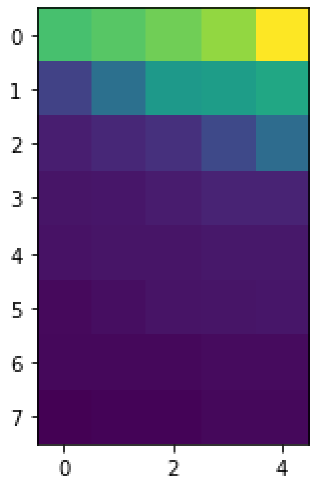
y_pred = dbscan.fit_predict(Cancer_X)

sum(y_pred==-1)

In [83]:

```
eps = [50, 100, 150, 200, 250, 300, 400, 500] #10 &100
min_samples =[10, 15, 20, 25, 30]
noise=np.zeros((8,5))
for i in range(8):
    for j in range(5):
        dbscan= DBSCAN(eps=eps[i], min_samples=min_samples[j])
        y_pred = dbscan.fit_predict(Cancer_X)
        noise[i][j]=sum(y_pred==-1)
print(noise)
plt.imshow(noise)
plt.show()
```

```
[[187. 195. 206. 220. 262.]
 [ 56. 100. 143. 148. 158.]
 [ 27.  34.  41.  62.  96.]
 [ 20.  21.  25.  31.  31.]
 [ 18.  20.  20.  22.  22.]
 [ 12.  15.  19.  20.  21.]
 [ 11.  11.  11.  13.  13.]
 [  5.   8.   8.  11.  11.]]
```



Question 8j: What observations can you make about the clustering structure in this data, based on the matrices you generated for 8(g) and 8(h)?

Answer: To better understand the data, I have increased the range of values for eps and minsamples ,I infer that the points are widely distrubted in high dimensional space. when EPS =50 and min samples is 10 a cluster has been formed with 382 points and 187 points as noise points.This only 382 points were densly located and the remaining points are widely spread in the high dimensional space.

Question 8k: Select the parameters for eps, min_samples based on your answers for 8(g), 8(h) and 8(i). Compute cluster assigments using DBSCAN. Compute RandIndex of the cluster assignments with respect to the true labels.

In [84]:

```
dbscan= DBSCAN(eps=100, min_samples=10) #100 10
y_pred = dbscan.fit_predict(Cancer_X)
r=np.where(y_pred==-1, 100, y_pred)
score=rand_index(r,Cancer_y)
print(score)
```

```
0.6680115844451595
```

Question 8l: Compare RandIndex from 8(g) with that of 8(k) and determine which algorithm performed best? Based on this, comment on how the data/clusters may be distributed in R^d.

Answer: DBSCAN algorithm works better when compared with K means ,the points in the clusters were widely distributed but at each location close to each other ,the density among the points at those locations is very high.

In [85]:

```
k=hopkins(Cancer_X)
print(k)
#print(type(k))
```

```
0.9632856607400602
```