Hands-on Exercise for CLUS Module

0. Setting up necessary packages and creating data

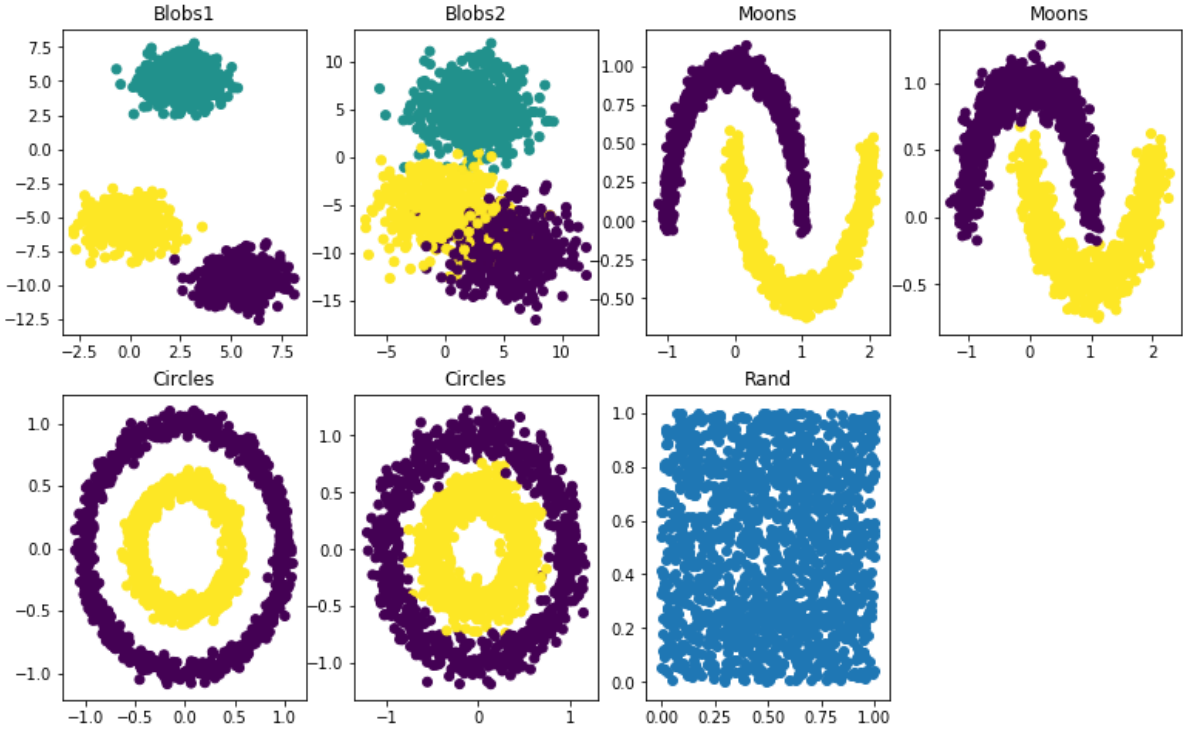
In [2]:
```
!pip install --user scikit-learn --upgrade
```

```
Requirement already up-to-date: scikit-learn in ./.local/lib/python3.6/site-p
ackages
Requirement already up-to-date: scipy>=0.17.0 in ./.local/lib/python3.6/site-
packages (from scikit-learn)
Requirement already up-to-date: joblib>=0.11 in ./.local/lib/python3.6/site-p
ackages (from scikit-learn)
Requirement already up-to-date: numpy>=1.11.0 in ./.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,facecolor=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_lo
wer for next 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.

Blobs. K-Means work well for the clusters where mean of the data points is the center of the cluster as K-Means aims at minimising SSE and blobs represent the same.

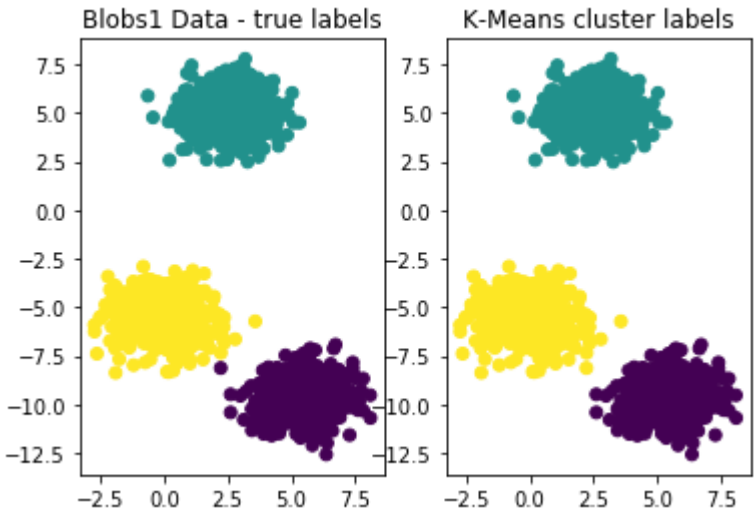
Question 1b: Without running K-Means clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where k-Means is expected to NOT work well. Support your answer by explaining your rationale.

Except blobs, K-Means fails for the rest of datasets. As mentioned in 1a, K-Means work well for clusters where mean of the data points is concentrated at the center of the cluster and hence it fails for Moons dataset. It can't handle circles data because the there are two circular clusters with different radius centered at the same mean and the means are not well seperated.

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.

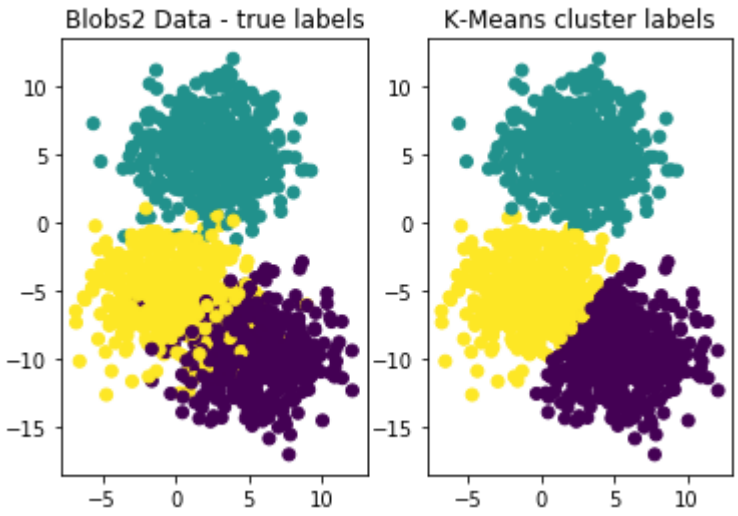
In [8]:
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Blobs1 = kmeans.fit_predict(Blobs1_X)
```

In [9]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y) # true clusters
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred_Blobs1)  # KMeans cluster
s
plt.title('K-Means cluster labels')
plt.show()
```



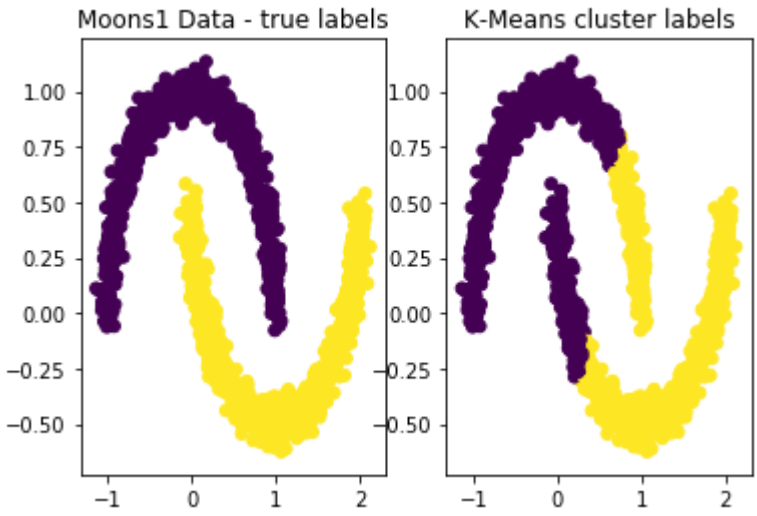
In [10]:
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Blobs2 = kmeans.fit_predict(Blobs2_X)
```

In [11]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs1_y) # true clusters
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred_Blobs2)  # KMeans cluster
s
plt.title('K-Means cluster labels')
plt.show()
```



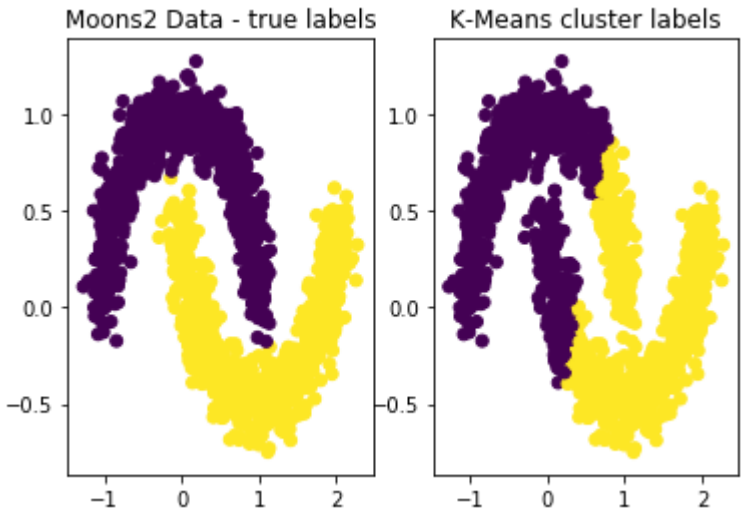
In [12]:
```
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Moons1 = kmeans.fit_predict(Moons1_X)
```

In [13]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y) # true clusters
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred_Moons1)  # KMeans cluster
s
plt.title('K-Means cluster labels')
plt.show()
```



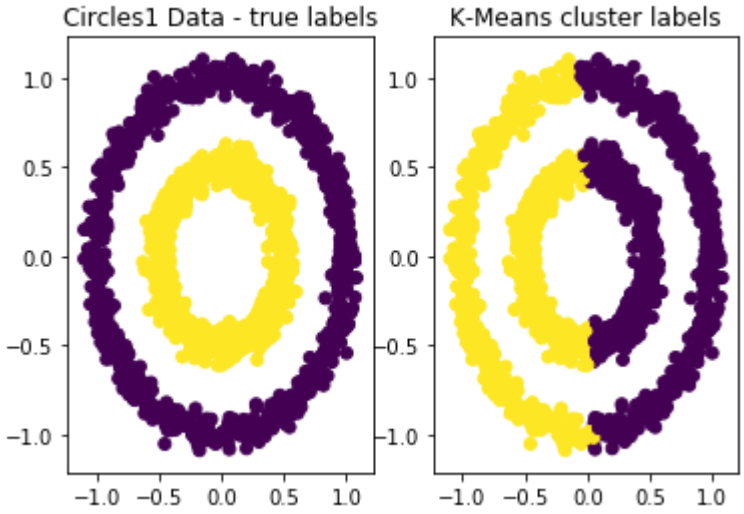
In [14]:
```
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Moons2 = kmeans.fit_predict(Moons2_X)
```

In [15]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y) # true clusters
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred_Moons2)  # KMeans cluster
s
plt.title('K-Means cluster labels')
plt.show()
```



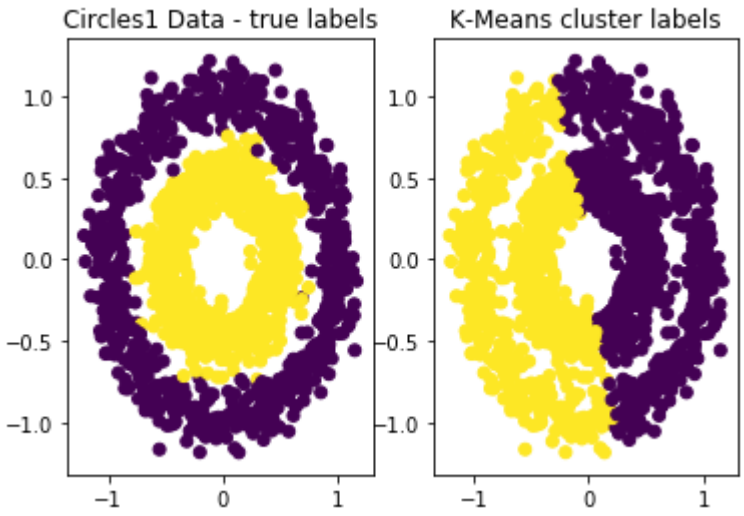
In [16]:
```
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Circles1 = kmeans.fit_predict(Circles1_X)
```

In [17]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y) # true clusters
plt.title('Circles1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred_Circles1)  # KMeans c
lusters
plt.title('K-Means cluster labels')
plt.show()
```



In [18]:
```
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_Circles2 = kmeans.fit_predict(Circles2_X)
```

In [19]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y) # true clusters
plt.title('Circles1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred_Circles2)  # KMeans c
lusters
plt.title('K-Means cluster labels')
plt.show()
```



Blobs>Moons=Circles. K-Means work well only for circular clusters where mean of the data points is the center of the cluster

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 [20]:
```
print(rand_index(y_pred_Blobs1,Blobs1_y))
print(rand_index(y_pred_Blobs2,Blobs2_y))
print(rand_index(y_pred_Moons1,Moons1_y))
print(rand_index(y_pred_Moons2,Moons2_y))
print(rand_index(y_pred_Circles1,Circles1_y))
print(rand_index(y_pred_Circles2,Circles2_y))
```

```
0.99911140760507
0.9207142539470758
0.6201236379808761
0.6240836112964199
0.4996744496330887
0.4996806760062264
```

Blobs1>Blobs2>Moons1=Moons2>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.

Yes.

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.

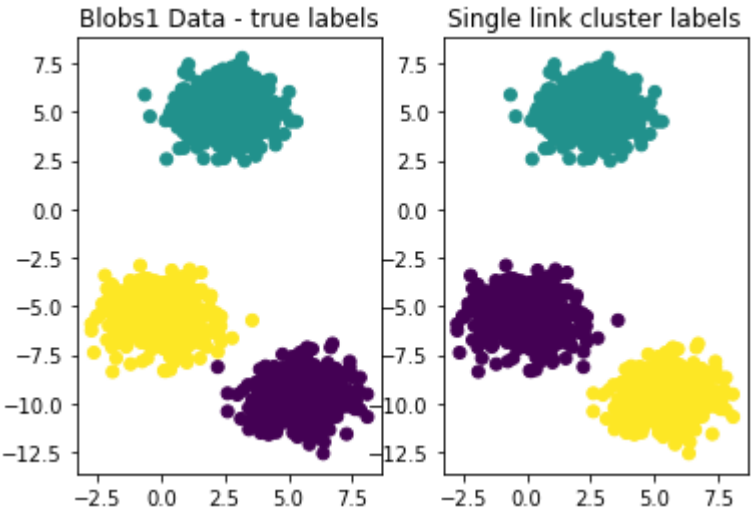
Single link works well for Blobs1, Moons1, Circles1. Single link tries to group the points in such a way that smallest minimum pairwise distance is achieved among the clusters i.e it chooses the clusters in a way that most similar points(having less distance among them) are grouped together.

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.

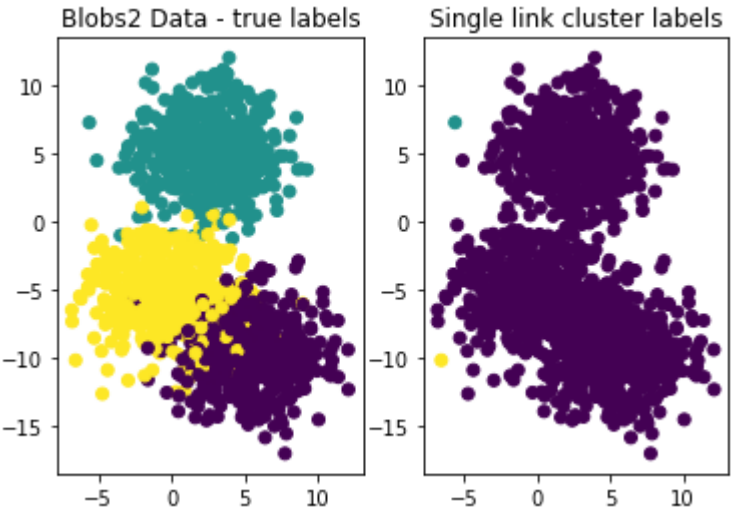
It doesn't work well for Blob2, Moons2, Circles2. In Blobs2, Moons2, Circles2 since the inter cluster distance is very less, Single link clusters the whole set of data points as a single cluster.

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.

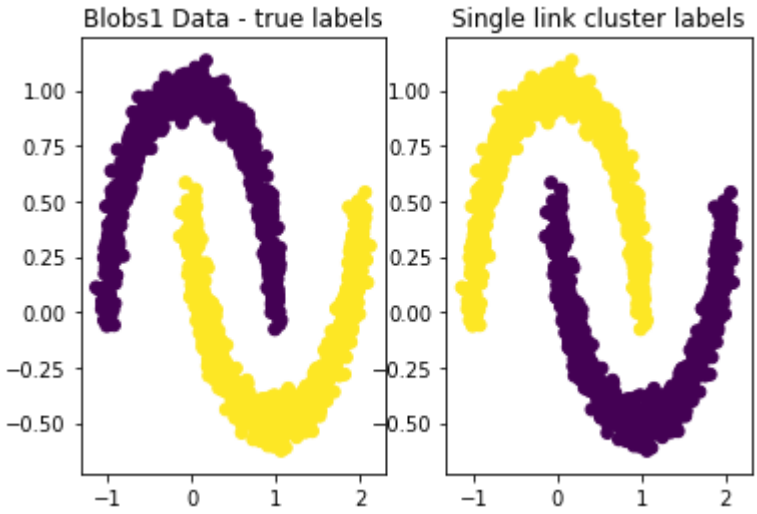
In [21]:
```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_blobs1 = single_linkage.fit_predict(Blobs1_X)
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred_single_blobs1)
plt.title('Single link cluster labels')
plt.show()
```



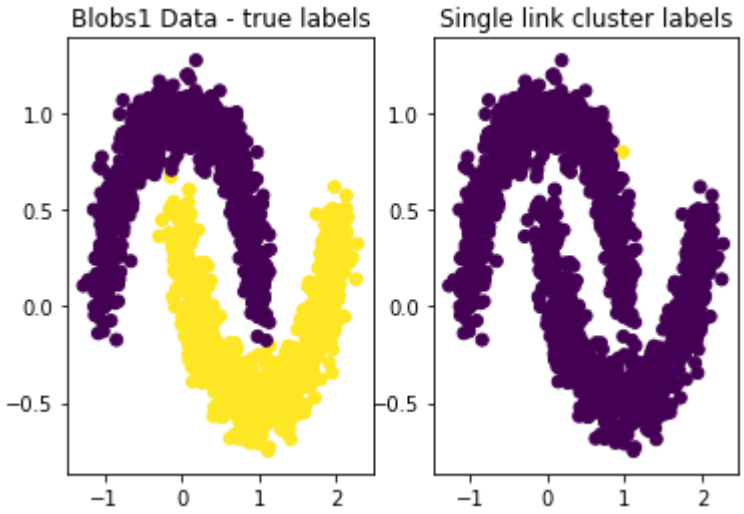
In [22]:
```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_Blobs2 = single_linkage.fit_predict(Blobs2_X)
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred_single_Blobs2)
plt.title('Single link cluster labels')
plt.show()
```



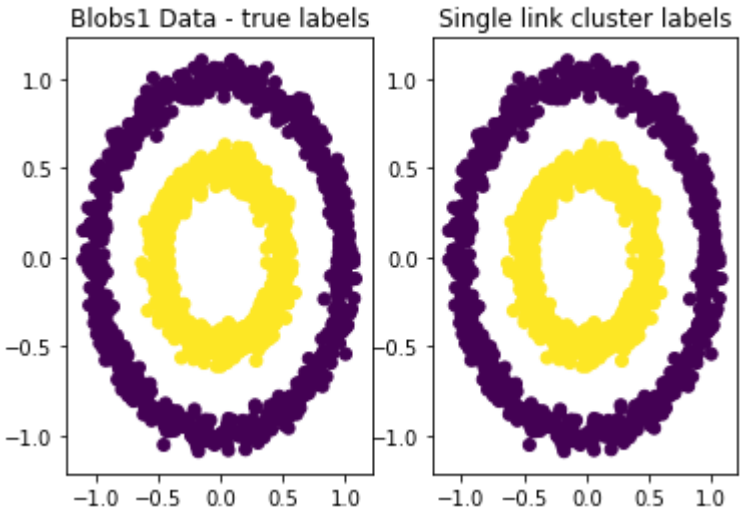
In [23]:
```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_Moons1 = single_linkage.fit_predict(Moons1_X)
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred_single_Moons1)
plt.title('Single link cluster labels')
plt.show()
```



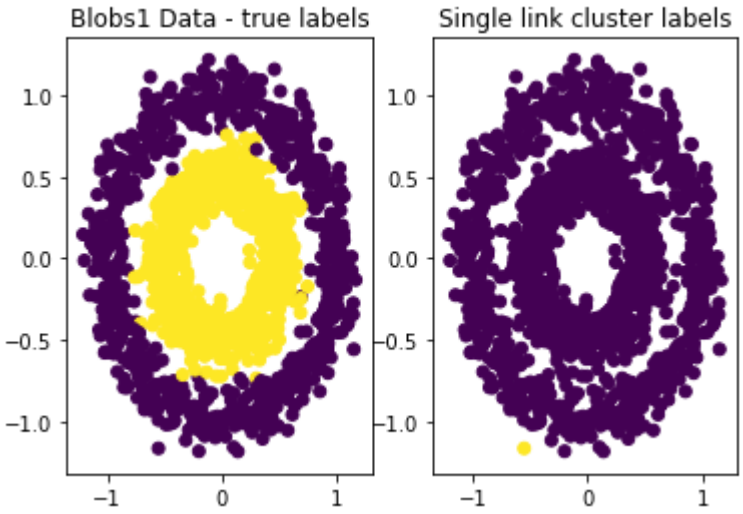
In [24]:
```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_Moons2 = single_linkage.fit_predict(Moons2_X)
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred_single_Moons2)
plt.title('Single link cluster labels')
plt.show()
```



In [25]:
```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_Circles1 = single_linkage.fit_predict(Circles1_X)
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred_single_Circles1)
plt.title('Single link cluster labels')
plt.show()
```



In [26]:
```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single_Circles2 = single_linkage.fit_predict(Circles2_X)
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred_single_Circles2)
plt.title('Single link cluster labels')
plt.show()
```



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 [27]:
```
print(rand_index(y_pred_single_blobs1,Blobs1_y))
print(rand_index(y_pred_single_Blobs2,Blobs2_y))
print(rand_index(y_pred_single_Moons1,Moons1_y))
print(rand_index(y_pred_single_Moons2,Moons2_y))
print(rand_index(y_pred_single_Circles1,Circles1_y))
print(rand_index(y_pred_single_Circles2,Circles2_y))
```

```
0.99911140760507
0.33377896375361354
1.0
0.49966733377807426
1.0
0.49966733377807426
```

Moons1=Circles1=Blobs1>Moons2=Circles2>Blobs2

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.

Yes.

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.

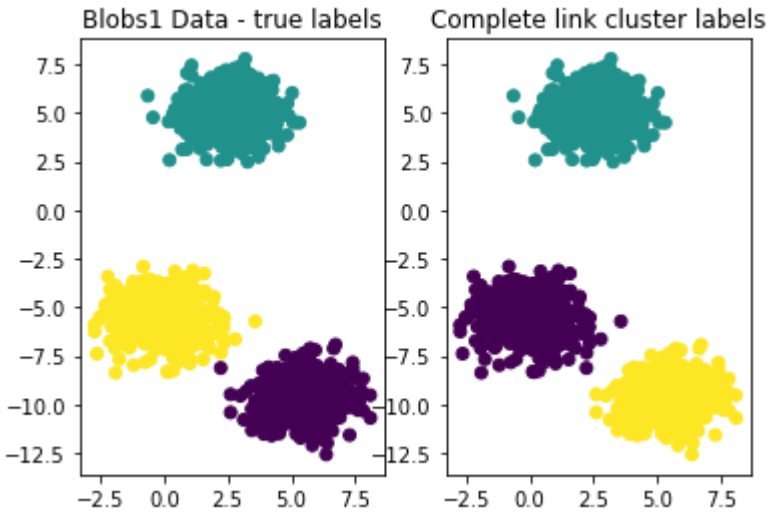
Max Link performs well for Blobs1. Max link tries to group the points in such a way that smallest maximum pairwise distance is achieved among the clusters i.e most distant points(having maximum distance from one another) also exhibit appropriate similarity in a cluster and we observe that in Blobs1.

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.

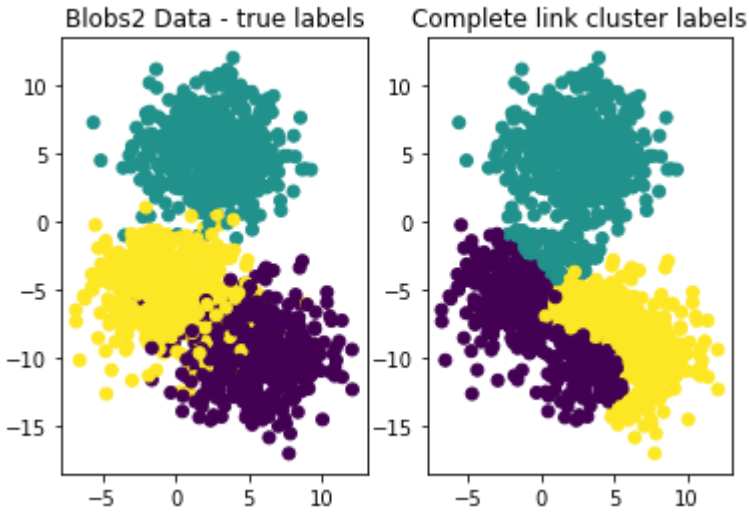
Blobs2, Moons1, Moons2, circles1, circles2. As already mentioned in 3a, Max-Link tries to maximise the similarity between the points which are of maximum distance to each other in a cluster and in these datasets, the points in one cluster which appears close by to that of the other cluster, it tries to group all those points in a single cluster.

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.

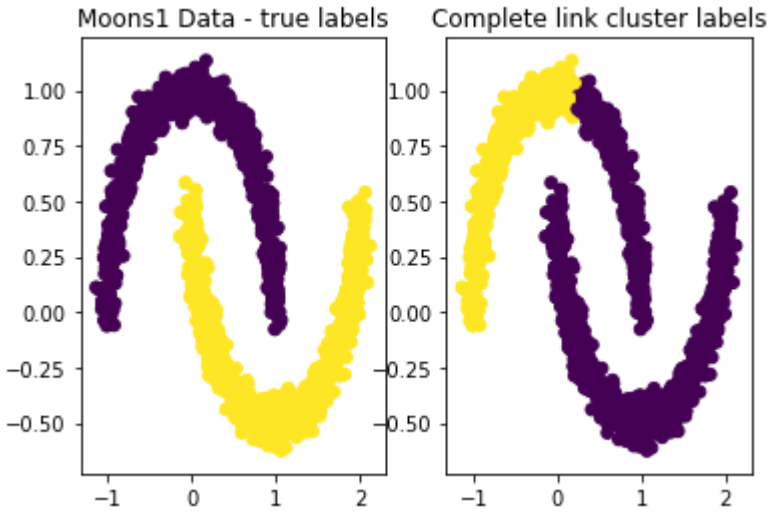
In [28]:
```
n_clusters = 3
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Blobs1 = complete_linkage.fit_predict(Blobs1_X)
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred_com_Blobs1)
plt.title('Complete link cluster labels')
plt.show()
```



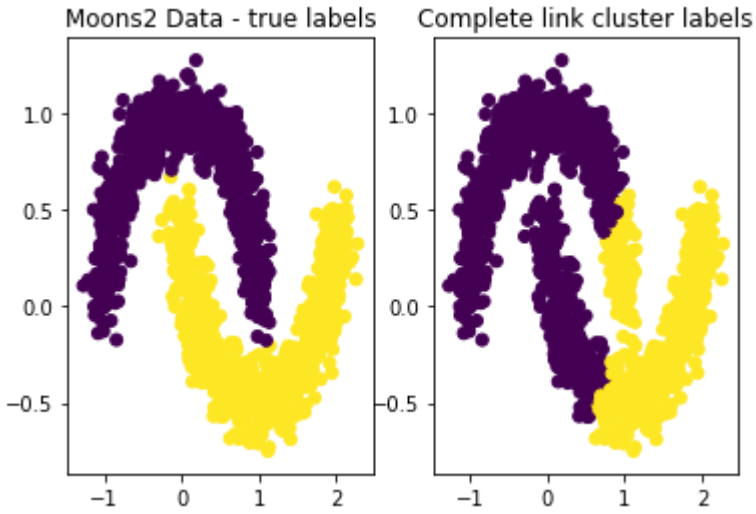
In [29]:
```
n_clusters = 3
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Blobs2 = complete_linkage.fit_predict(Blobs2_X)
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred_com_Blobs2)

plt.title('Complete link cluster labels')
plt.show()
```



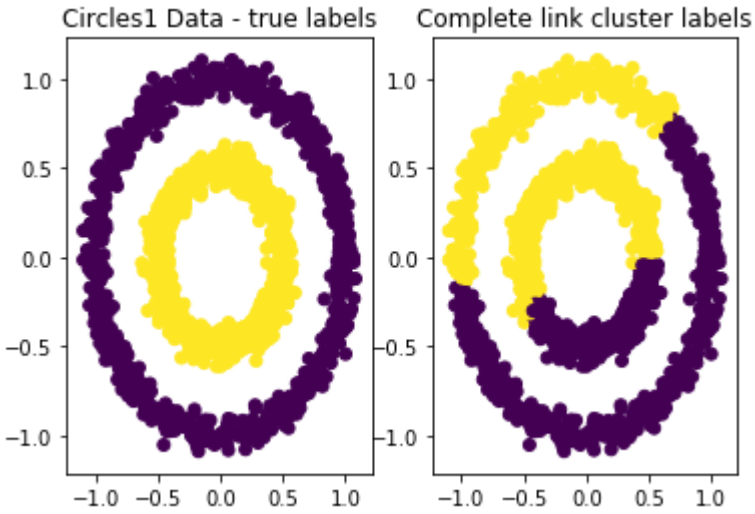
In [30]:
```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Moons1 = complete_linkage.fit_predict(Moons1_X)
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred_com_Moons1)
plt.title('Complete link cluster labels')
plt.show()
```



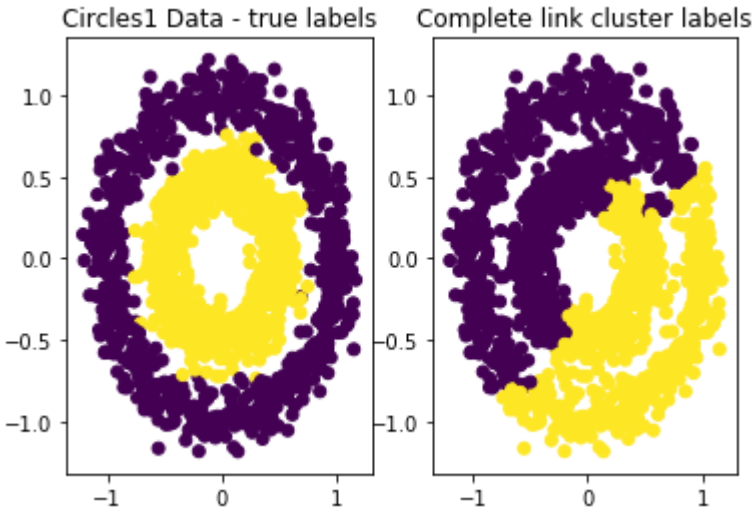
In [31]:
```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Moons2 = complete_linkage.fit_predict(Moons2_X)
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred_com_Moons2)
plt.title('Complete link cluster labels')
plt.show()
```



In [32]:
```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Circles1 = complete_linkage.fit_predict(Circles1_X)
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=y_pred_com_Circles1)
plt.title('Complete link cluster labels')
plt.show()
```



In [33]:
```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
usters)
y_pred_com_Circles2 = complete_linkage.fit_predict(Circles2_X)
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred_com_Circles2)
plt.title('Complete link cluster labels')
plt.show()
```



Blobs1>Blobs2=Moons1=Moons2=Circles1=Circles2

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 [34]:
```
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)
```

In [35]:
```
print(rand_index(y_pred_com_Blobs1,Blobs1_y))
print(rand_index(y_pred_com_Blobs2,Blobs2_y))
print(rand_index(y_pred_com_Moons1,Moons1_y))
print(rand_index(y_pred_com_Moons2,Moons2_y))
print(rand_index(y_pred_com_Circles1,Circles1_y))
print(rand_index(y_pred_com_Circles2,Circles2_y))
```

```
0.99911140760507
0.7736544362908606
0.662605292417167
0.5965310206804536
0.5218714698688014
0.5000587058038692
```

Blobs1>Blobs2>Moons1>Moons2>Circles1>Circles2

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.

Yes

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.

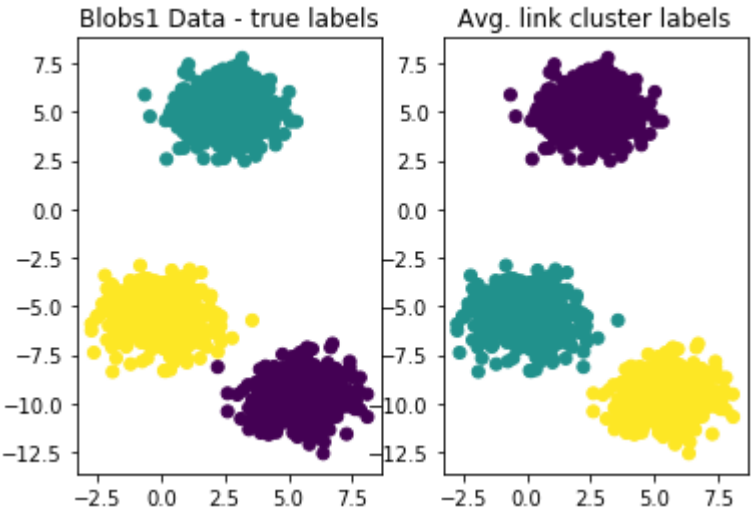
Blobs1. Average link aims at maximising the intra cluster similarity the pair of clusters with the highest cohesion and hence the individual cluster have to be dense and wel seperated from the others which is observed in Blobs1.

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.

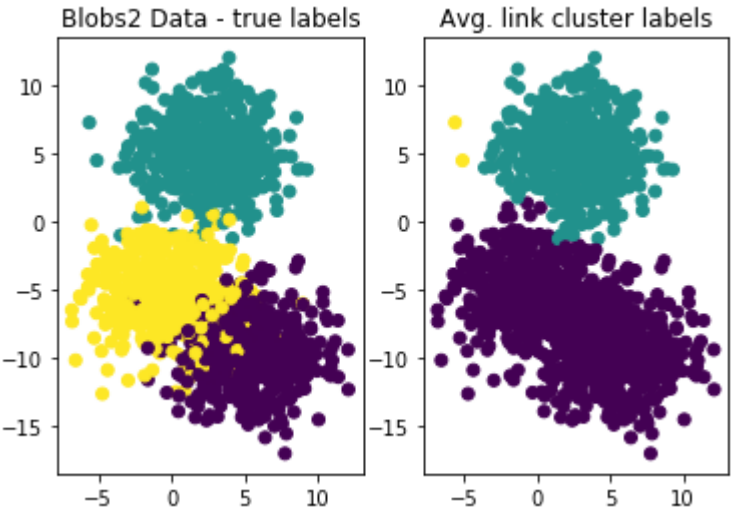
Except Blobs1 it fails for the other datasets since the clusters are not well seperated and Average link tries to group the nearest points(which may not belong to the same cluster) to achieve high cohesion.

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.

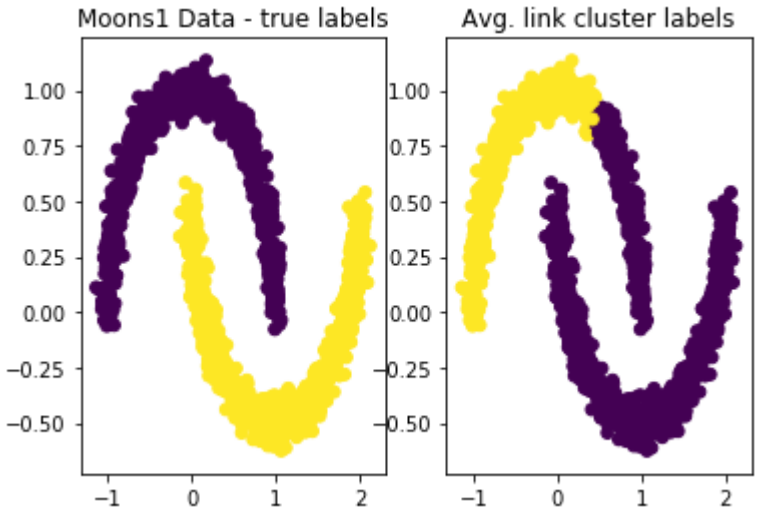
In [36]:
```
n_clusters = 3
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Blobs1 = average_linkage.fit_predict(Blobs1_X)
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred_avg_Blobs1)
plt.title('Avg. link cluster labels')
plt.show()
```



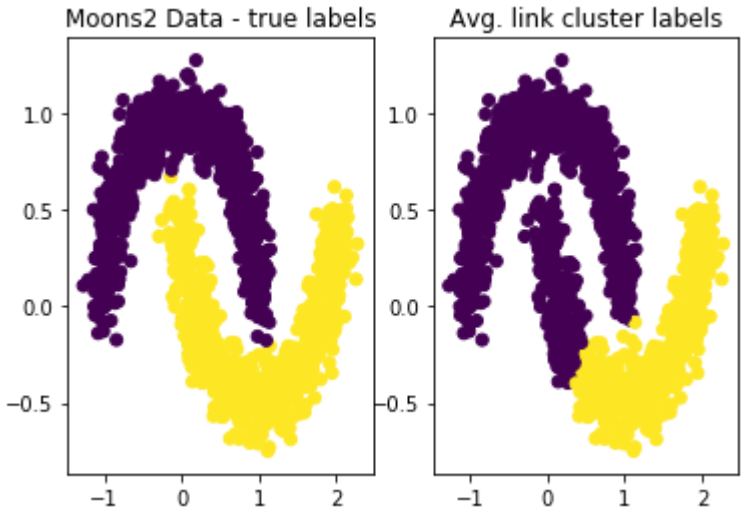
In [37]:
```
n_clusters = 3
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Blobs2 = average_linkage.fit_predict(Blobs2_X)
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred_avg_Blobs2)
plt.title('Avg. link cluster labels')
plt.show()
```



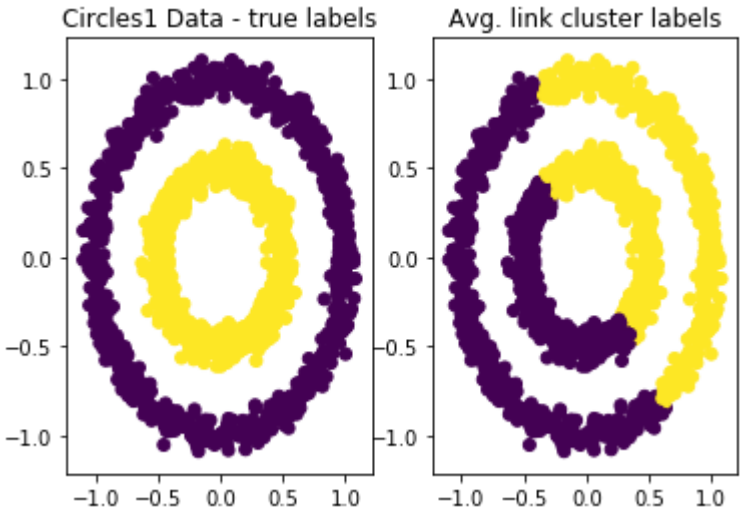
In [38]:
```
n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Moons1 = average_linkage.fit_predict(Moons1_X)
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred_avg_Moons1)
plt.title('Avg. link cluster labels')
plt.show()
```



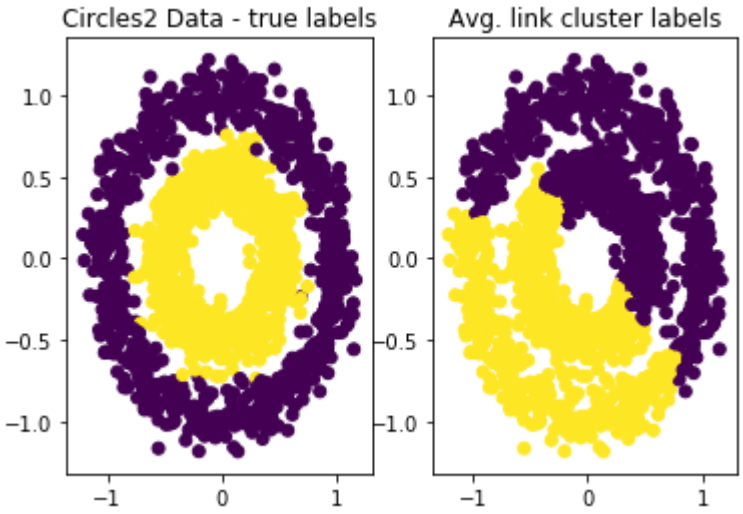
In [39]:
```
n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Moons2 = average_linkage.fit_predict(Moons2_X)
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred_avg_Moons2)
plt.title('Avg. link cluster labels')
plt.show()
```



In [40]:
```
n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Circles1 = average_linkage.fit_predict(Circles1_X)
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=y_pred_avg_Circles1)
plt.title('Avg. link cluster labels')
plt.show()
```



In [41]:
```
n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clus
ters)
y_pred_avg_Circles2 = average_linkage.fit_predict(Circles2_X)
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=y_pred_avg_Circles2)
plt.title('Avg. link cluster labels')
plt.show()
```



Blobs1>Blobs2>Moons1=Moons2>Circles1=Circles2

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 [42]:
```
print(rand_index(y_pred_avg_Blobs1,Blobs1_y))
print(rand_index(y_pred_avg_Blobs2,Blobs2_y))
print(rand_index(y_pred_avg_Moons1,Moons1_y))
print(rand_index(y_pred_avg_Moons2,Moons2_y))
print(rand_index(y_pred_avg_Circles1,Circles1_y))
print(rand_index(y_pred_avg_Circles2,Circles2_y))
```

```
0.99911140760507
0.7636575494774294
0.7132310429175005
0.7457647320435846
0.500414498554592
0.5050780520346898
```

Blobs1>Blobs2>Moons2>Moons1>Circles1=Circles2

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.

Yes

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.

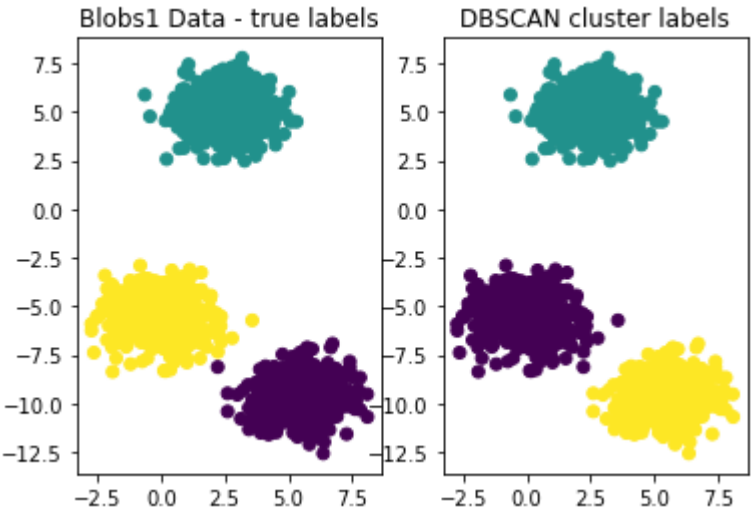
DBScan performs well for Blobs1, Moons and Circles. DBScan tries to join all the core points making it a cluster. It completely depends upon the epsilon and min_points defined for the algorithm. In these datasets, DBScan wud be able to identify the core points in such a way that the resultant clusters are in accordance with the original clusters(There might be noise points in Moons2 and Circles2 which are dependent upon the min_points paramter).

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.

Blobs2. DBScan performs well when cluster is dense and noise is sparse. Blobs2 is more likely circular shaped overlapping clusters and DBScan tries to identify the core points along the borders and couldn't cluster well.

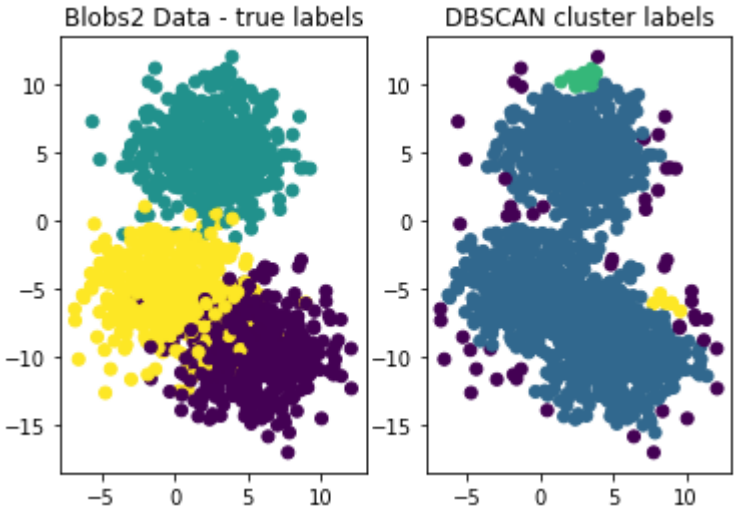
Question 5c: Run DBSCAN clustering algorithm on all the datasets (except Rand). **Choose eps and min_samples parameters to make sure that DBSCAN finds the same number of clusters as in the ground truth ('Data_y').** Visualize the clusters for each of them. Based on the visualization, rank the datasets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

In [43]:
```
dbscan = DBSCAN(eps=1.4, min_samples=10)
y_pred_dbs_Blobs1 = dbscan.fit_predict(Blobs1_X)
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=y_pred_dbs_Blobs1)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Blobs1==-1)
```



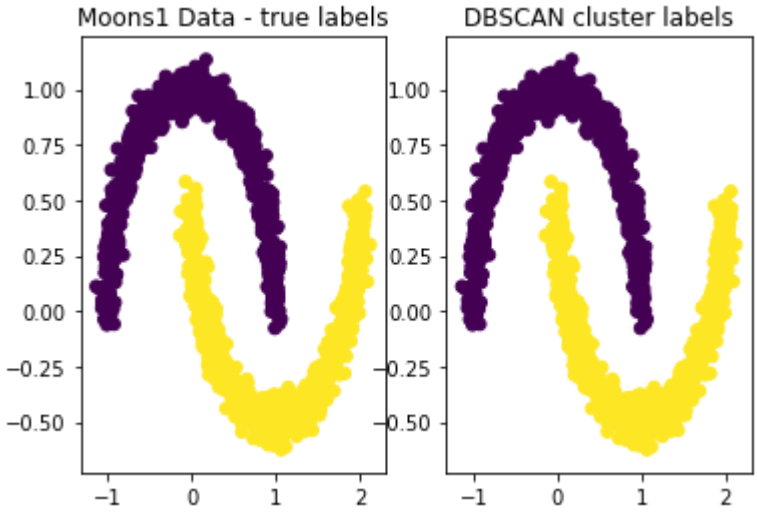
Out[43]: 0

In [44]:
```
dbscan = DBSCAN(eps=0.87, min_samples=4)
y_pred_dbs_Blobs2 = dbscan.fit_predict(Blobs2_X)
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=y_pred_dbs_Blobs2)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Blobs2==-1)
```



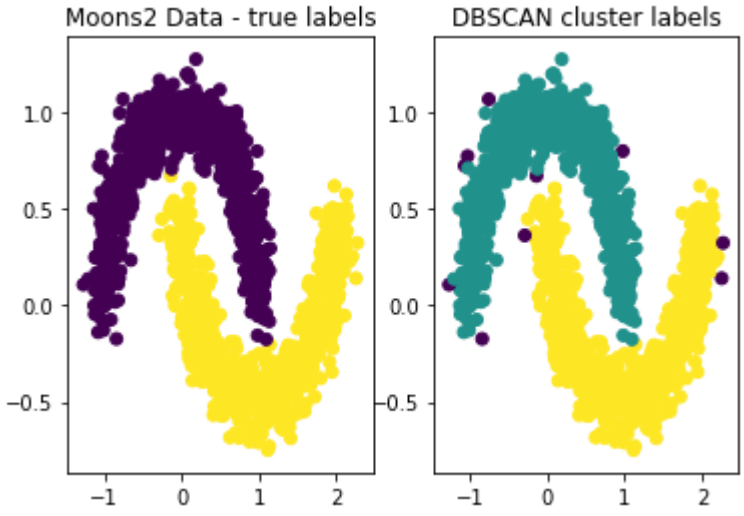
Out[44]: 53

In [45]:
```
dbscan = DBSCAN(eps=0.2, min_samples=30)
y_pred_dbs_Moons1 = dbscan.fit_predict(Moons1_X)
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=y_pred_dbs_Moons1)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Moons1==-1)
```



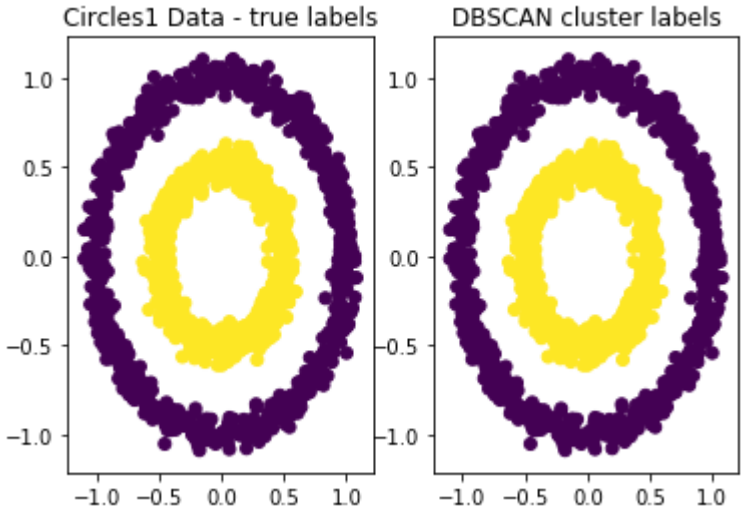
Out[45]: 0

In [46]:
```
dbscan = DBSCAN(eps=0.131, min_samples=10)
y_pred_dbs_Moons2 = dbscan.fit_predict(Moons2_X)
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=y_pred_dbs_Moons2)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Moons2==-1)
```



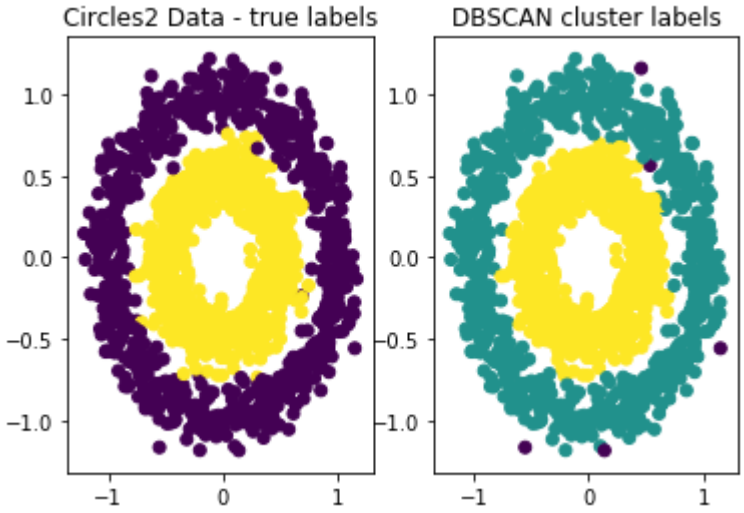
Out[46]: 10

In [47]:
```
dbscan = DBSCAN(eps=0.131, min_samples=10)
y_pred_dbs_Circles1 = dbscan.fit_predict(Circles1_X)
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=y_pred_dbs_Circles1)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Circles1==-1)
```



Out[47]: 0

In [48]:
```
dbscan = DBSCAN(eps=0.15, min_samples=20)
y_pred_dbs_Circles2 = dbscan.fit_predict(Circles2_X)
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=y_pred_dbs_Circles2)
plt.title('DBSCAN cluster labels')
plt.show()
np.sum(y_pred_dbs_Circles2==-1)
```



Out[48]: 5

Blobs1=Moons1=Circles1>>Moons2>Circles>Blobs2

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?

Noise points are depicted along with plots in 5c. Blobs1, Moons1, Circles1 have 0 noise points since these datasets are dense, have least deviation among the points and well clustered without any overlapping.

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 [49]:
```
for i in range (0,y_pred_dbs_Blobs2.shape[0]):
    if y_pred_dbs_Blobs2[i]==-1 :
        y_pred_dbs_Blobs2[i]=2

for i in range (0,y_pred_dbs_Moons2.shape[0]):
    if y_pred_dbs_Moons2[i]==-1 :
        y_pred_dbs_Moons2[i]=2

for i in range (0,y_pred_dbs_Circles2.shape[0]):
    if y_pred_dbs_Circles2[i]==-1 :
        y_pred_dbs_Circles2[i]=2
```

In [50]:
```
from scipy.special import comb
def rand_index_count(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)
```

In [51]:
```
print(rand_index_count(y_pred_dbs_Blobs1,Blobs1_y))
print(rand_index_count(y_pred_dbs_Blobs2,Blobs2_y))
print(rand_index_count(y_pred_dbs_Moons1,Moons1_y))
print(rand_index_count(y_pred_dbs_Moons2,Moons2_y))
print(rand_index_count(y_pred_dbs_Circles1,Circles1_y))
print(rand_index_count(y_pred_dbs_Circles2,Circles2_y))
```

```
0.99911140760507
0.36475783855903937
1.0
0.9933537914164999
1.0
0.9769401823437848
```

Moons1=Circles1=Blobs1>Moons2>Circles2>Blobs2

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.

Yes

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.

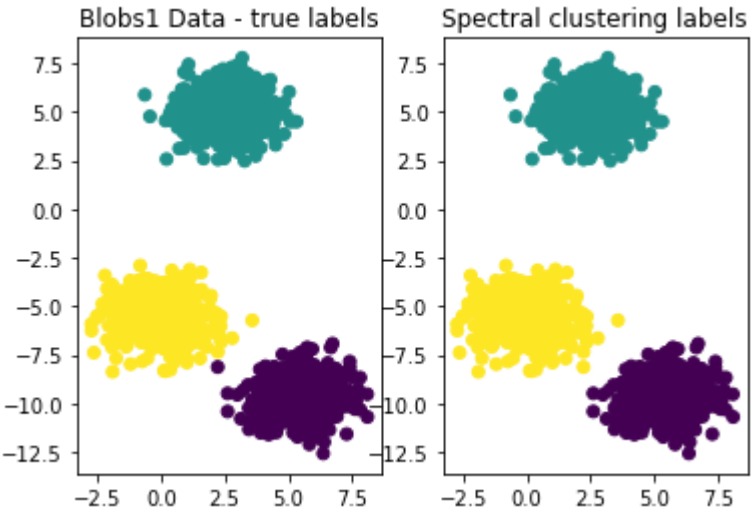
All the datasets except Moons2, Circles2. Spectral Clustering performs transformation of Affinity/Laplacian matrix into Eigen space and then perform K-Means to cluster the data. Blobs1 is well seperated and globular in shape and hence K-Means work well. Blobs2, Moons1, Circles1 when transformed into Eigen space, they are expected to have high cluster tendency and can be well clustered by K-Means.

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.

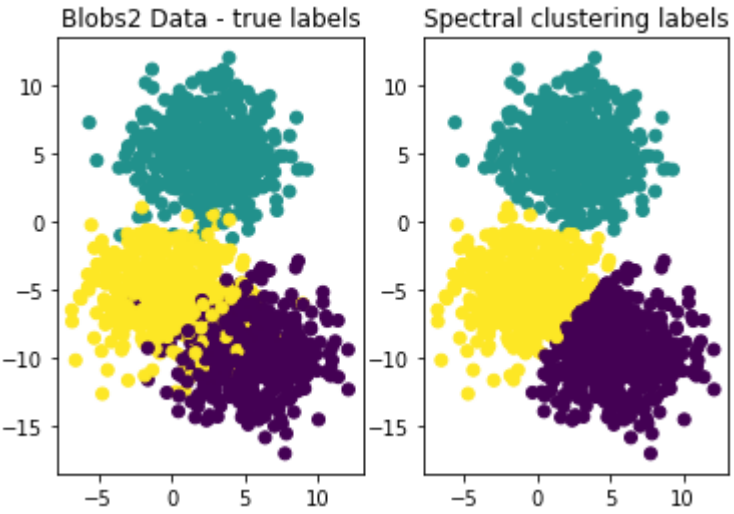
Moons2, Circles2. Even after performing Laplacian transformation Moons2, Circles2 can't have high cluster tendency due to the very least inter cluster distance between points and K-Means fails to identify clusters as per the true labels.

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.

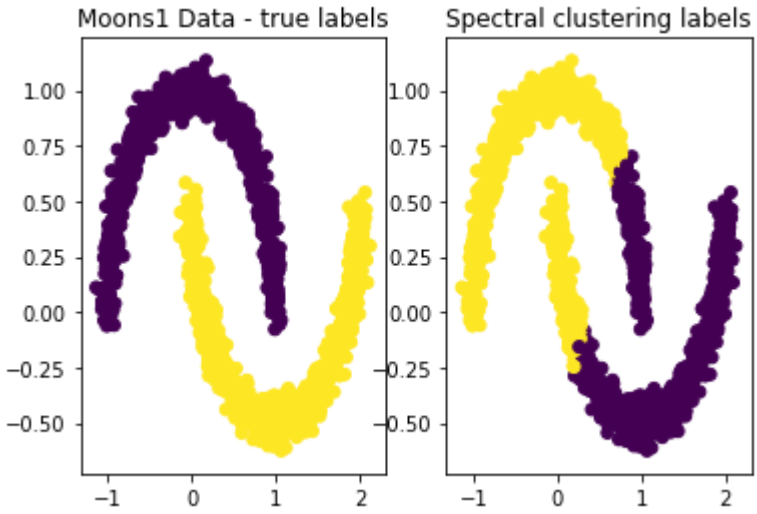
In [52]:
```
n_clusters = 3
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Blobs1 = spectral.fit_predict(Blobs1_X)
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred_sc_Blobs1)
plt.title('Spectral clustering labels')
plt.show()
```



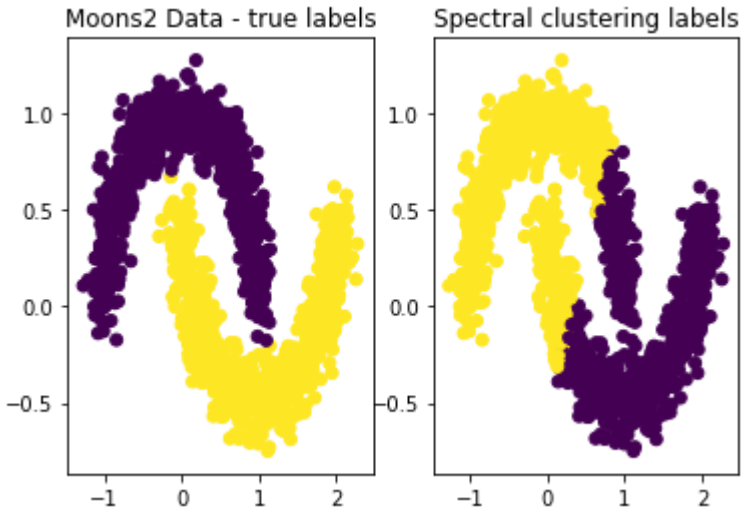
In [53]:
```
n_clusters = 3
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Blobs2 = spectral.fit_predict(Blobs2_X)
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred_sc_Blobs2)
plt.title('Spectral clustering labels')
plt.show()
```



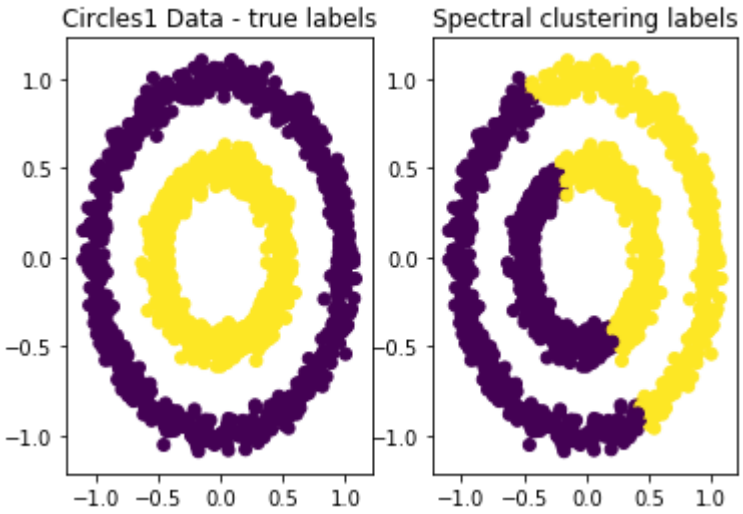
In [54]:
```
n_clusters = 2
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Moons1 = spectral.fit_predict(Moons1_X)
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred_sc_Moons1)
plt.title('Spectral clustering labels')
plt.show()
```



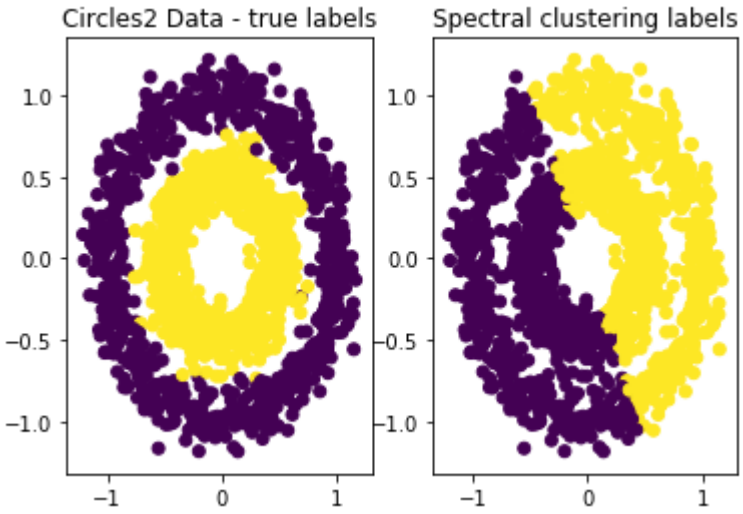
In [55]:
```
n_clusters = 2
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Moons2 = spectral.fit_predict(Moons2_X)
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred_sc_Moons2)
plt.title('Spectral clustering labels')
plt.show()
```



In [56]:
```
n_clusters = 2
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Circles1 = spectral.fit_predict(Circles1_X)
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=y_pred_sc_Circles1)
plt.title('Spectral clustering labels')
plt.show()
```



In [57]:
```
n_clusters = 2
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc_Circles2 = spectral.fit_predict(Circles2_X)
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=y_pred_sc_Circles2)
plt.title('Spectral clustering labels')
plt.show()
```



Blobs1>Blobs2>Moons1=Moons2>Circles1=Circles2

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 [58]:
```
print(rand_index_count(y_pred_sc_Blobs1,Blobs1_y))
print(rand_index_count(y_pred_sc_Blobs2,Blobs2_y))
print(rand_index_count(y_pred_sc_Moons1,Moons1_y))
print(rand_index_count(y_pred_sc_Moons2,Moons2_y))
print(rand_index_count(y_pred_sc_Circles1,Circles1_y))
print(rand_index_count(y_pred_sc_Circles2,Circles2_y))
```

```
0.99911140760507
0.919189682010229
0.6441263064265066
0.6448441183010896
0.49966733377807426
0.4997553924838781
```

Blobs1>Blobs2>Moons1=Moons2>Circles1=Circles2

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.

Yes. But my rationale in 6a and 6b are not in accordance with 6c and 6d because by default the affinity parameter will be rbf kernel in SpectralClustering. My rationale is per the nearest neighbour.

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.

Blobs1 followed by Moons1 and Circles1. In these datasets, Intra cluster is having high cohesion and inter cluster is having high adhesion which states good clustering tendency.

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.

Rand, Circles2 followed by Moons2 and Blobs2. Rand doesn't have any predefined structure of the points to be clustered and it's just a dataset with random points. Circles2 doesn't exhibit good cluster tendency as we observe overlapping of datapoints in two clusters.

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

In [59]:
```
print("Blobs1 " + str(hopkins(Blobs1_X)))
print("Blobs2 " + str(hopkins(Blobs2_X)))
print("Moons1 " + str(hopkins(Moons1_X)))
print("Moons2 " + str(hopkins(Moons2_X)))
print("Circles1 " + str(hopkins(Circles1_X)))
print("Circles2 " + str(hopkins(Circles2_X)))
print("Rand " + str(hopkins(Rand_X)))
```

```
Blobs1 0.9370325680037499
Blobs2 0.8305143505968465
Moons1 0.9216245818294703
Moons2 0.8991778962978844
Circles1 0.8261528623548402
Circles2 0.7720757973884986
Rand 0.6053031980976354
```

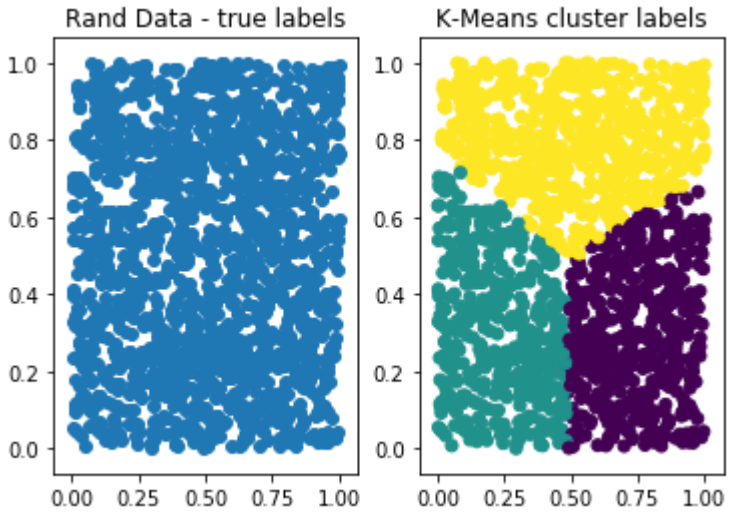
Blobs1>Moons1>Moons2>Blobs2>Circles1>Circles2>Random

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

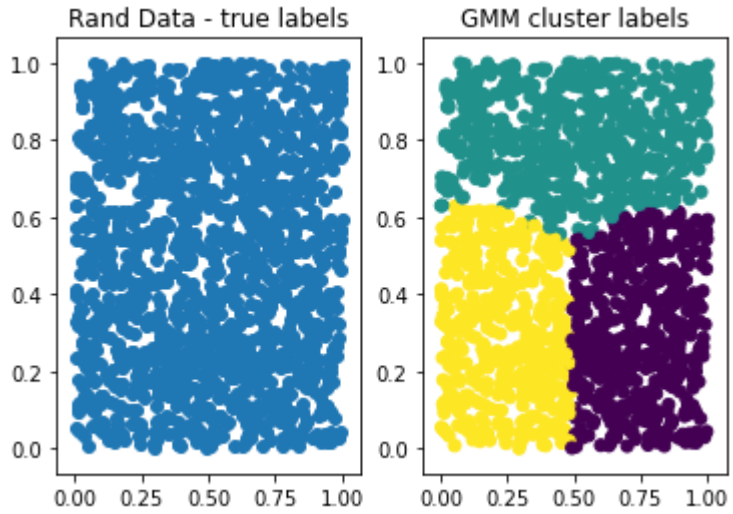
No. I expected circles1 to have the similar stat with Moons1 but Circles1 is having less tendency. Hopkin's statistic fails to measure the tendency for concentric circles due to the closeness in the 2 circles.

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.

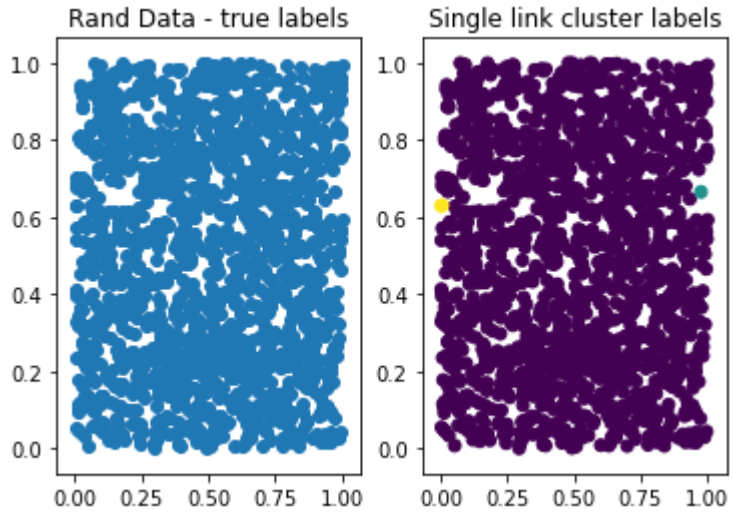
In [60]:
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred_km = kmeans.fit_predict(Rand_X)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1]) # true clusters
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_km)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```



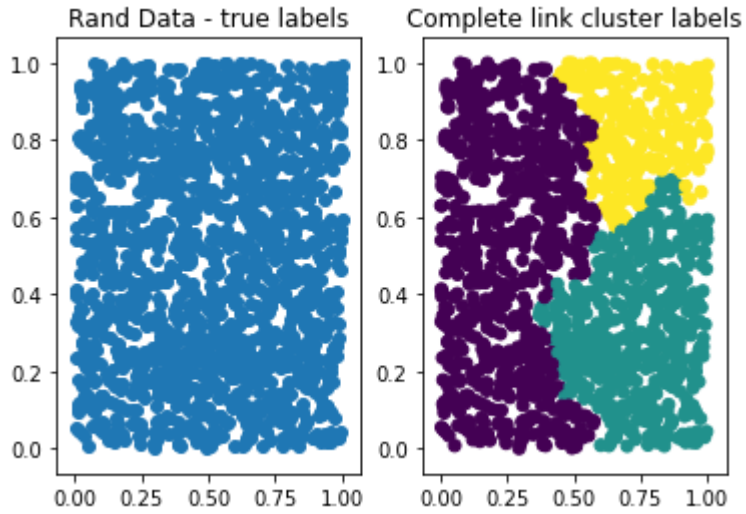
In [61]:
```
n_clusters = 3;
gmm = GaussianMixture(n_components=n_clusters, covariance_type='full')
y_pred_gm = gmm.fit_predict(Rand_X)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1]) # true clusters
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_gm)  # KMeans clusters
plt.title('GMM cluster labels')
plt.show()
```



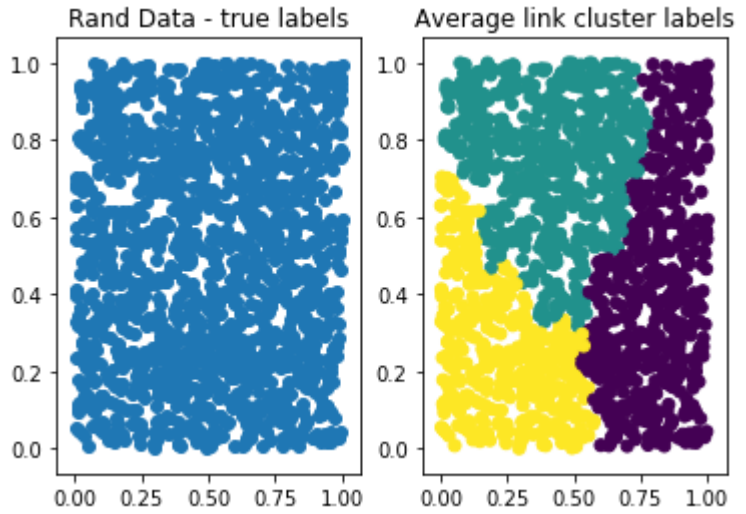
In [62]:
```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_cluste
rs)
y_pred_single = single_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_single)
plt.title('Single link cluster labels')
plt.show()
```



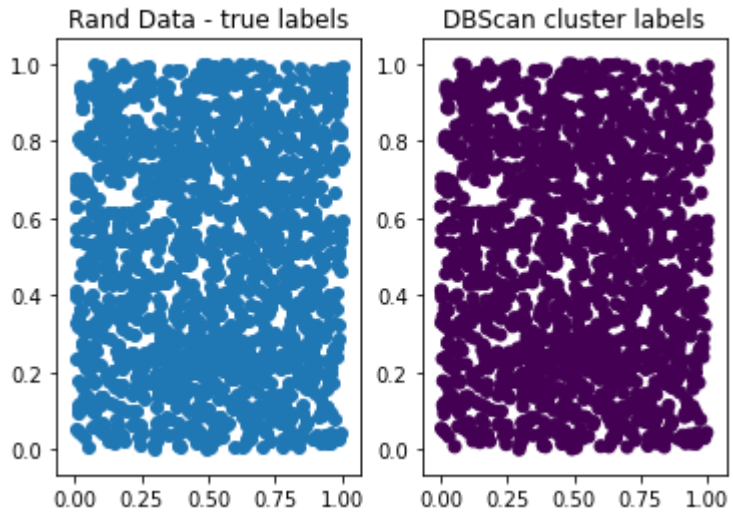
In [63]:
```
n_clusters = 3
max_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cluster
s)
y_pred_max = max_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_max)
plt.title('Complete link cluster labels')
plt.show()
```



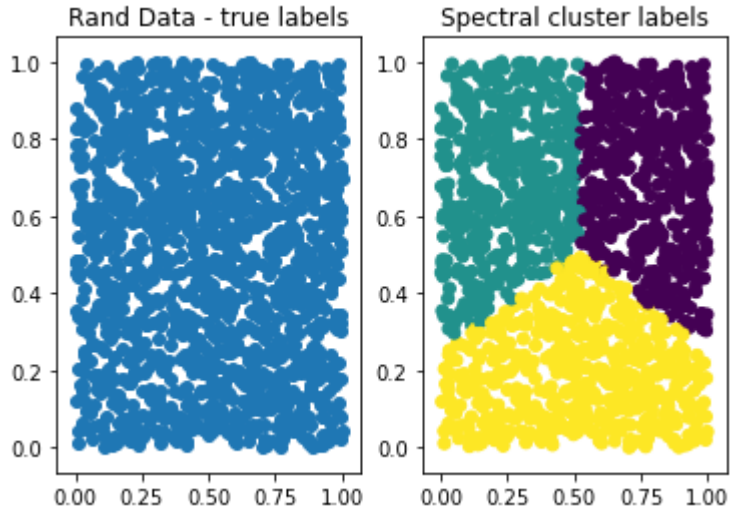
In [64]:
```
n_clusters = 3
Avg_linkage = AgglomerativeClustering(linkage="average", n_clusters=n_clusters
)
y_pred_avg = Avg_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_avg)
plt.title('Average link cluster labels')
plt.show()
```



In [65]:
```
dbscan = DBSCAN(eps=0.8, min_samples=10)
y_pred_dbs = dbscan.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_dbs)
plt.title('DBScan cluster labels')
plt.show()
```



In [141]:
```
n_clusters = 3
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state
)
y_pred_sc = spectral.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred_sc)
plt.title('Spectral cluster labels')
plt.show()
```



1) K-means : Cluster boundaries are governed by the fact that it minimises the SSE and hence it's more likely to be blobs shaped. 2) GMM : Each point is assigned to a cluster based upon the Normal distrubution(Probability is high at mean and decays as we move farther from mean) and hence we observe a bell shaped boundary. 3) Single-link : Single-link tries to achieve the smallest minimum pairwise distance and merges almost all data points to one cluster. 4) Complete link : Complete-link tries to achieve the smallest maximum pairwise distance and it keeps on grouping the points into different clusters. 5) Average link : Avg link tries tries to balance both single and max link clustering and hence we obtained 3 clusters. 6) DBScan : DBScan depends on the parameters eps and minsamples. In Rand dataset, we don't observe any structure in the data and for the parameters which I have chosen, it's merging the whole dataset as a single cluster. 7) Spectral Clustering : Data is transformed into Eigen Space such that the cluster tendency is improved and then we perform K-Means which gives the resultant clusters similar to K-Means for Rand dataset.

8. Real-world dataset

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

In [66]:
```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The features are:

In [67]:
```
cancer.feature_names
```

Out[67]:
```
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 [68]:
```
cancer.target_names
```

Out[68]:
```
array(['malignant', 'benign'], dtype='<U9')
```

Create dataset for classification

In [67]:
```
Cancer_X = cancer.data
Cancer_y = cancer.target
```

Size of Cancer_X and Cancer_y

In [70]:
```
Cancer_X.shape
```

Out[70]: (569, 30)

In [71]:
```
Cancer_y.shape
```

Out[71]: (569,)

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

In [72]:
```
for n_clusters in range (2,41):
    kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
    kmeans.fit_predict(Cancer_X)
    score = -(kmeans.score(Cancer_X))
    print("k = " + str(n_clusters) + " SSE : " + str(score))
```

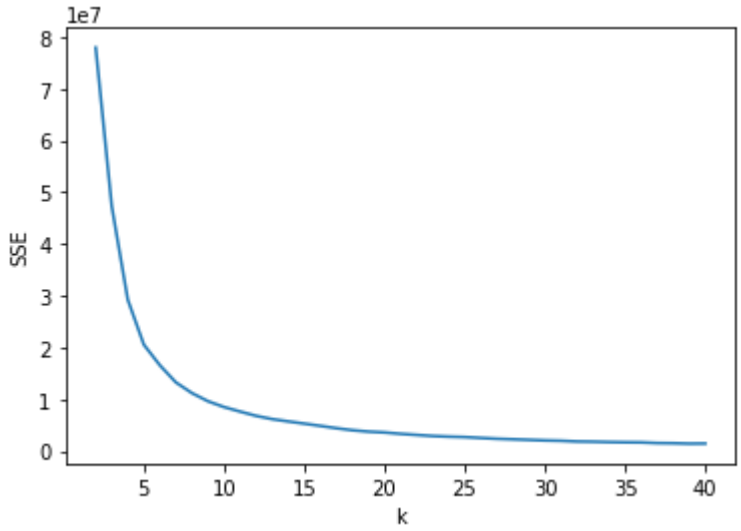
```
k = 2 SSE : 77943099.8782987
k = 3 SSE : 47285926.90370661
k = 4 SSE : 29226541.65197972
k = 5 SSE : 20539877.62210295
k = 6 SSE : 16558716.702017363
k = 7 SSE : 13249736.068326872
k = 8 SSE : 11183535.784621993
k = 9 SSE : 9609383.579294508
k = 10 SSE : 8487166.052065082
k = 11 SSE : 7613587.210442201
k = 12 SSE : 6784588.861542951
k = 13 SSE : 6157087.418915496
k = 14 SSE : 5708365.126097778
k = 15 SSE : 5286031.401124285
k = 16 SSE : 4848940.456318591
k = 17 SSE : 4398276.578374952
k = 18 SSE : 4009831.035727249
k = 19 SSE : 3738118.103648788
k = 20 SSE : 3578729.294742346
k = 21 SSE : 3312041.5895248177
k = 22 SSE : 3102392.224607761
k = 23 SSE : 2894387.689415504
k = 24 SSE : 2768624.6807041294
k = 25 SSE : 2685795.48332426
k = 26 SSE : 2514580.5049141482
k = 27 SSE : 2362959.9456619583
k = 28 SSE : 2257591.621312524
k = 29 SSE : 2148955.4860370243
k = 30 SSE : 2036763.9989706709
k = 31 SSE : 1969448.345201505
k = 32 SSE : 1833170.6268923914
k = 33 SSE : 1791368.9975101275
k = 34 SSE : 1722589.7579862447
k = 35 SSE : 1677340.1256733793
k = 36 SSE : 1656114.5396019104
k = 37 SSE : 1528956.6311424784
k = 38 SSE : 1496563.4519065977
k = 39 SSE : 1417439.0170133682
k = 40 SSE : 1435813.7725891466
```

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

In [73]:
```
sse = {}
for k in range(2, 41):
    kmeans = KMeans(n_clusters=k, random_state=random_state)
    kmeans.fit(Cancer_X)
    sse[k] = -(kmeans.score(Cancer_X))
print (sse)
```

```
{2: 77943099.8782987, 3: 47285926.90370661, 4: 29226541.65197972, 5: 2053987
7.62210295, 6: 16558716.702017363, 7: 13249736.068326872, 8: 11183535.7846219
93, 9: 9609383.579294508, 10: 8487166.052065082, 11: 7613587.210442201, 12: 6
784588.861542951, 13: 6157087.418915496, 14: 5708365.126097778, 15: 5286031.4
01124285, 16: 4848940.456318591, 17: 4398276.578374952, 18: 4009831.03572724
9, 19: 3738118.103648788, 20: 3578729.294742346, 21: 3312041.5895248177, 22:
3102392.224607761, 23: 2894387.689415504, 24: 2768624.6807041294, 25: 268579
5.48332426, 26: 2514580.5049141482, 27: 2362959.9456619583, 28: 2257591.62131
2524, 29: 2148955.4860370243, 30: 2036763.9989706709, 31: 1969448.345201505,
32: 1833170.6268923914, 33: 1791368.9975101275, 34: 1722589.7579862447, 35: 1
677340.1256733793, 36: 1656114.5396019104, 37: 1528956.6311424784, 38: 149656
3.4519065977, 39: 1417439.0170133682, 40: 1435813.7725891466}
```

In [74]:
```
plt.xlabel('k'); plt.ylabel('SSE')
x=list(sse.keys())
y=list(sse.values())
plt.plot(x,y)
plt.show()
```



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

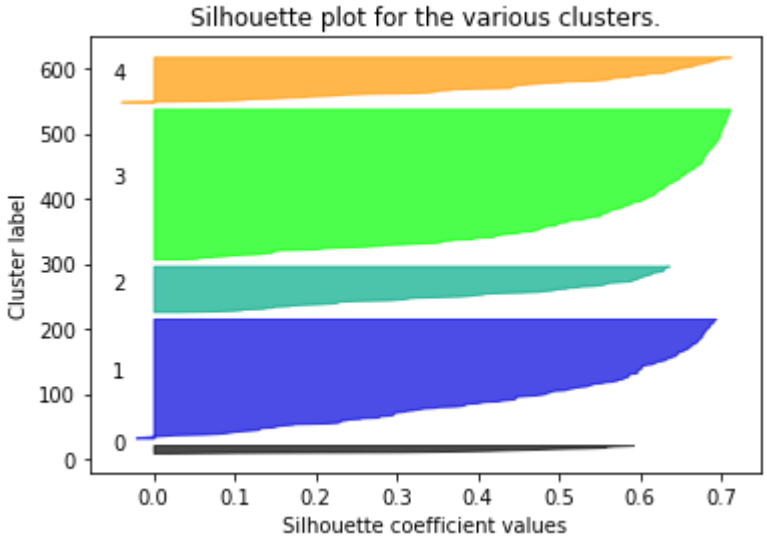
k=5 since for we observe the SSE measure being standardized from 5.

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

In [68]:
```
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred=kmeans.fit_predict(Cancer_X)
```

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

In [69]:
```
silhouette(Cancer_X,y_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?

Clustering tendency for the clusters is better for 1,3,4 and we treat them as good clusters when compared to 0,2

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 [70]:
```
print(rand_index_count(Cancer_y,y_pred))
```

```
0.6625473402807001
```

Quality of clusters is 66% which is not bad.

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:

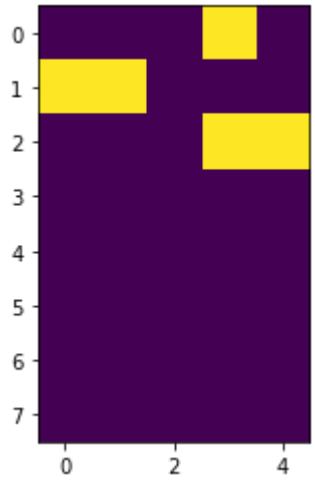
y_pred = dbscan.fit_predict(Cancer_X)

max(y_pred)+1

In [134]:
```
INIT_VALUE = "0"
COLUMN = 5
ROW = 8
def insert(matrix, num_row, num_col, value):
    matrix[num_row][num_col] = value
eps=[50, 100, 150, 200, 250, 300, 400, 500]
min_samples=[10, 15, 20, 25, 30]
matrix= [[INIT_VALUE for column in range(COLUMN)] for row in range(ROW)]
a=0
for i in eps:
    b=0
    for j in min_samples:
        dbscan = DBSCAN(eps=i, min_samples=j)
        y_pred_dbs = dbscan.fit_predict(Cancer_X)
        insert(matrix,a,b,(max(y_pred_dbs)+1))
        b=b+1
    a=a+1
for ele in matrix :
    print (ele)
```

```
[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]
```

In [129]:
```
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.imshow(matrix)
plt.show()
```



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.

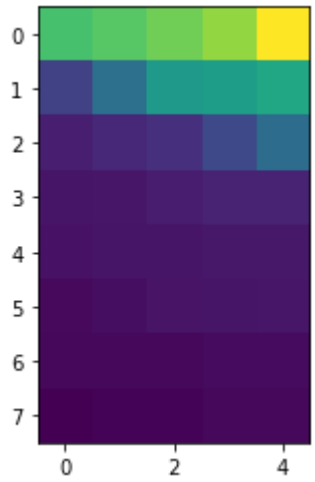
Hint: To compute the number of noise points, you may use:

y_pred = dbscan.fit_predict(Cancer_X)

sum(y_pred==-1)

In [135]:
```
#matrix= [[INIT_VALUE for column in range(COLUMN)] for row in range(ROW)]
eps=[50, 100, 150, 200, 250, 300, 400, 500]
min_samples=[10, 15, 20, 25, 30]
a=0
for i in eps:
    b=0
    for j in min_samples:
        dbscan = DBSCAN(eps=i, min_samples=j)
        y_pred_dbs = dbscan.fit_predict(Cancer_X)
        insert(matrix,a,b,sum(y_pred_dbs==-1))
        b=b+1
    a=a+1
for ele in matrix :
    print (ele)
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.imshow(matrix)
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)?

We observe that we arrive at desired number of clusters with comparitively less noise for the parameters eps=100, min_samples=10. To comment upon the structure, post clustering, we observe noise in the data which determines that the data is sparsely distributed.

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 [137]:
```
dbscan = DBSCAN(eps=100, min_samples=10)
y_pred_dbs_cancer = dbscan.fit_predict(Cancer_X)

for i in range (0,y_pred_dbs_cancer.shape[0]):
    if y_pred_dbs_cancer[i]==-1 :
        y_pred_dbs_cancer[i]=2
print(rand_index_count(Cancer_y,y_pred_dbs_cancer))
```

```
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.

DBSCAN is a bit performing better for the parameters(Eps,Minpts) which I have chosen. Clusters may be distributed arbitrarily without any globular shape(If they are K-Means wud have performed better) and they might be dense with good cohesion(Since DBScan performed better).