Hands-on Exercise for CLUS Module

November 28, 2021

1 Hands-on Exercise for CLUS Module

1.0.1 0. Setting up necessary packages and creating data

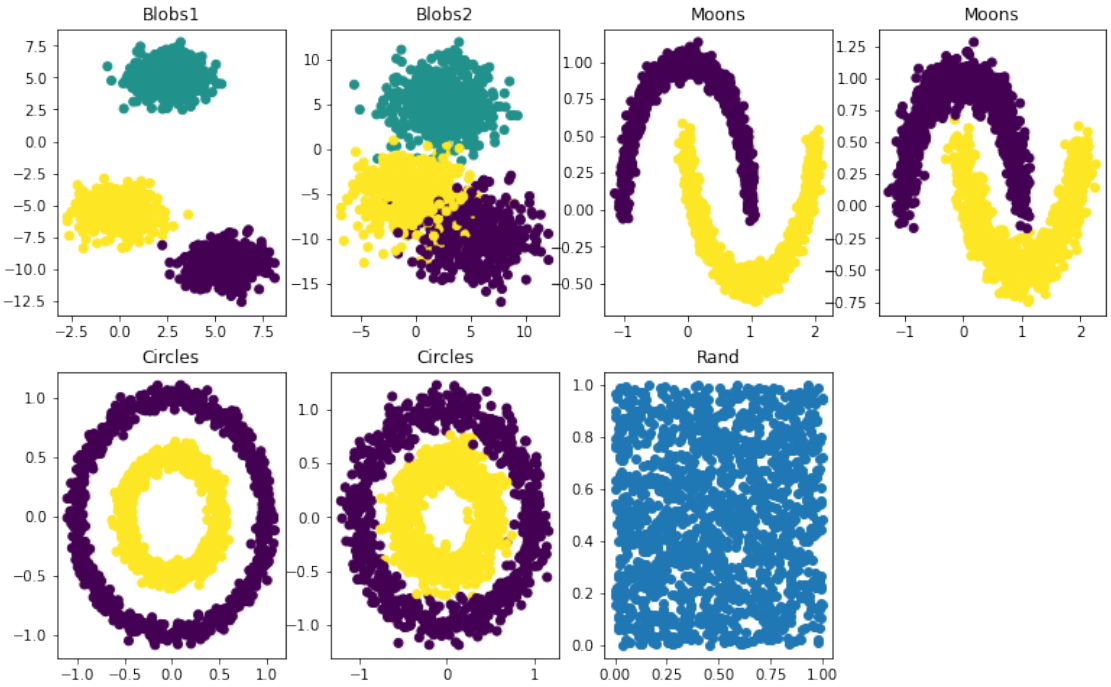
[1]:
```
!pip install --user scikit-learn --upgrade
```

```
Requirement already up-to-date: scikit-learn in ./.local/lib/python3.6/site-
packages
Requirement already up-to-date: scipy>=0.19.1 in ./.local/lib/python3.6/site-
packages (from scikit-learn)
Requirement already up-to-date: threadpoolctl>=2.0.0 in
./.local/lib/python3.6/site-packages (from scikit-learn)
Requirement already up-to-date: numpy>=1.13.3 in ./.local/lib/python3.6/site-
packages (from scikit-learn)
Requirement already up-to-date: joblib>=0.11 in ./.local/lib/python3.6/site-
packages (from scikit-learn)
You are using pip version 9.0.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
```

Import necessary packages

[2]:
```
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
```

[3]:
```
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

[4]:
```
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

[5]:
```
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

[6]:
```
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,alp
↪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 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()
```

1.0.2 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.

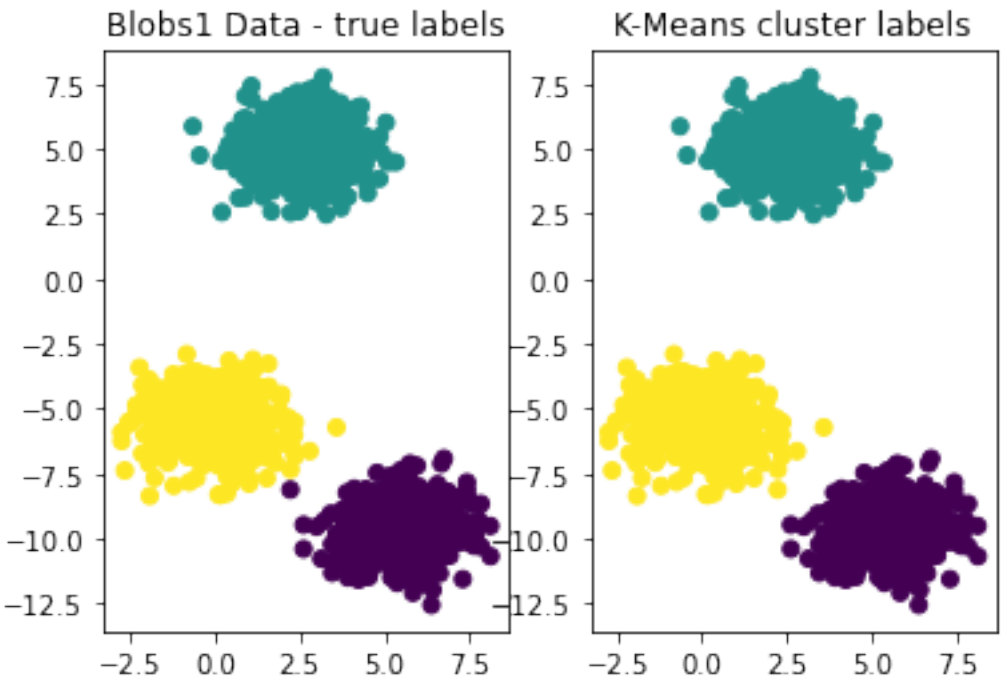
Answer: As K-means is dependent on SSE calculation and number of clusters, and if we take number of clusters as 3, then K-Means clustering will perfectly seperate the Blobs1 data as it is globular in shape and also the density of points seems to be pretty much same. Also, it has perfect cohesion and seperation. Blob2 will not be seperated that well as Blobs1 as the data has high standard deviation, which increases the distance from mean for points of the same cluster and also some points are overlapping with points in different clusters.

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.

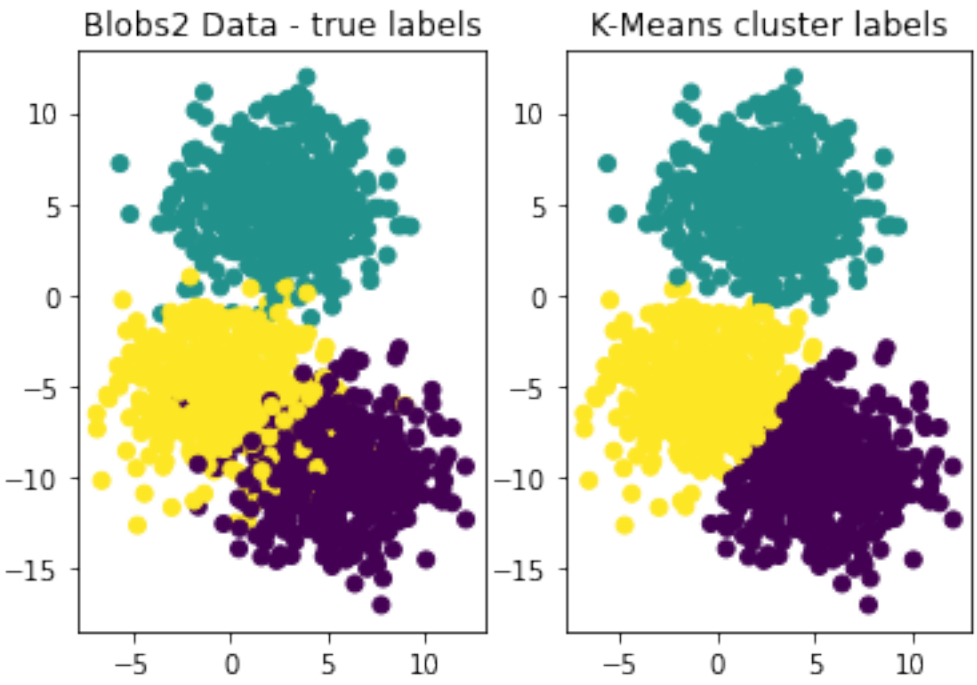
Answer: K-Means is not expected to work well in datasets where the density of points is high or clusters are of different sizes and non-globular. So, we can see that dataset Circles and Moons are non-globular, hence K-Means will not work well there when compared with Blobs dataset.

[7]:
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred1 = kmeans.fit_predict(Blobs1_X)
y_pred2 = kmeans.fit_predict(Blobs2_X)
```

[8]:
```
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_pred1)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```

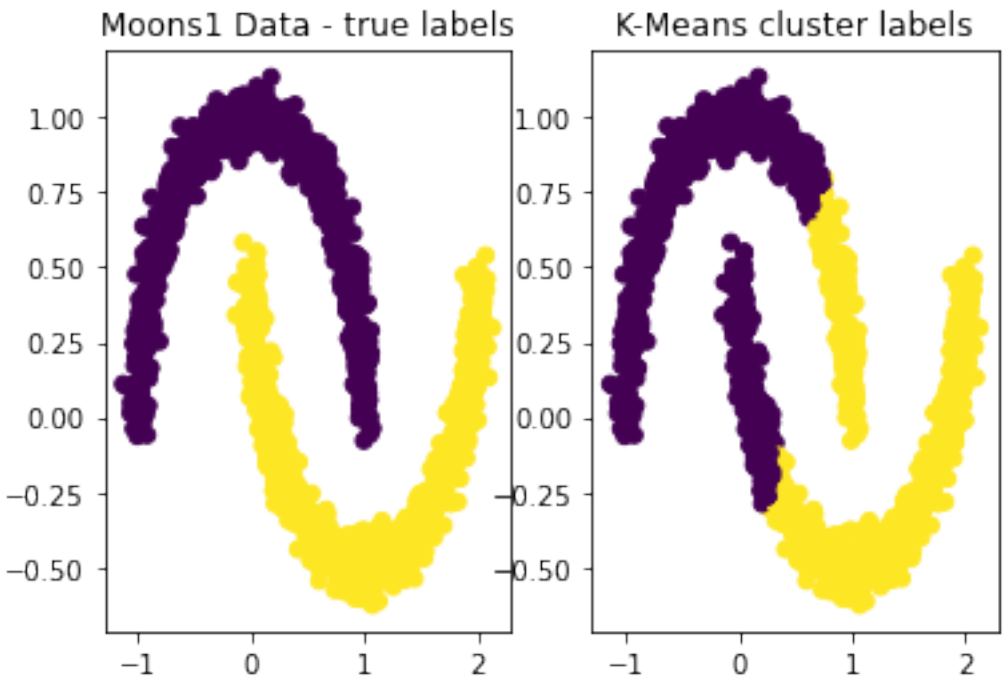


[9]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y) # true clusters
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred2)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```

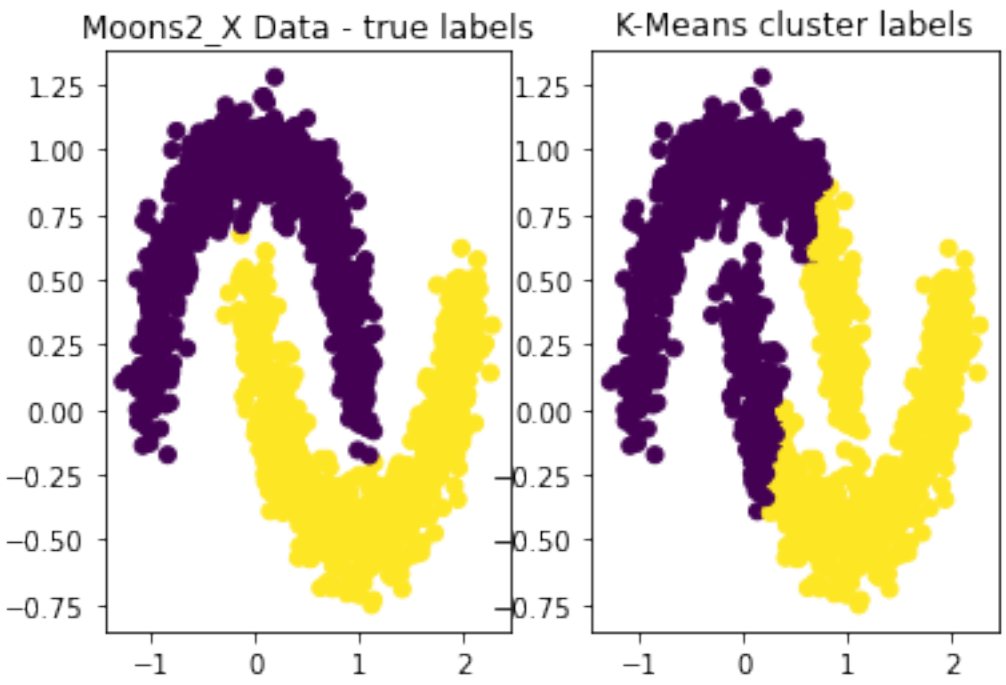


[10]:
```
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred3 = kmeans.fit_predict(Moons1_X)
y_pred4 = kmeans.fit_predict(Moons2_X)
y_pred5 = kmeans.fit_predict(Circles1_X)
y_pred6 = kmeans.fit_predict(Circles2_X)
```

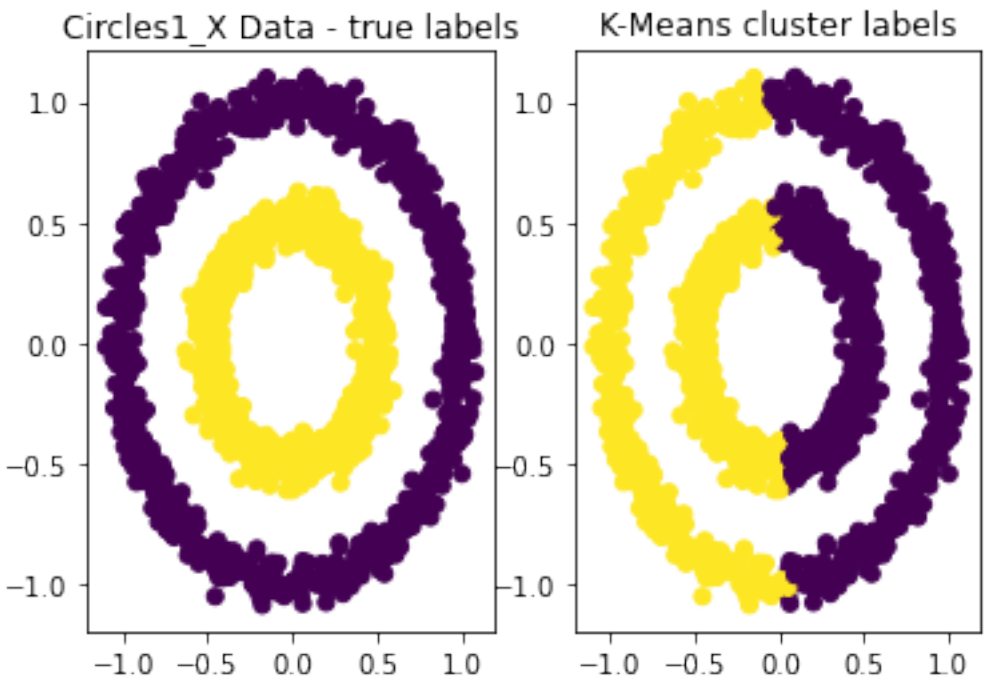
[11]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y) # true clusters
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred3)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```



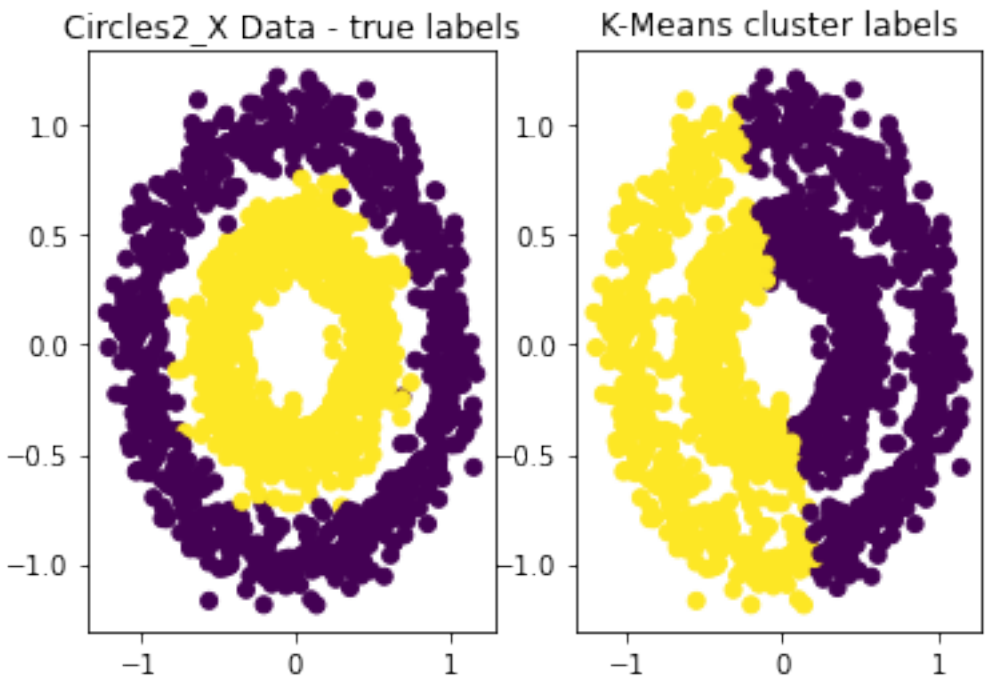
[12]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y) # true clusters
plt.title('Moons2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred4)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```



[13]:
```
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_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred5)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```



[14]:
```
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y) # true clusters
plt.title('Circles2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred6)  # KMeans clusters
plt.title('K-Means cluster labels')
plt.show()
```



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.

Answer: Ranking in decreasing order based on visulized performance: Blobs1, Blobs2, Moons2, Circles2, Moons1, Circles1. K-means works well on well seperated clusters which are non-globular in shape with similar sizes and densities. Blob1s seems to be perfect case for above mentioned conditions, hence it is ranked the highest in the order. Blobs2 follows it because of the high standard deviation from the mean of the clusters and also overlapping points between different clusters. Then comes the non-globular datasets Moon2 and Moons1 which are high in densities followed by Circles1 and Circles2.

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.

[15]:
```
print("Rand Index Value for Blobs1: ", rand_index(y_pred1, Blobs1_y))
print("Rand Index Value for Blobs2: ", rand_index(y_pred2, Blobs2_y))
print("Rand Index Value for Moons1: ", rand_index(y_pred3, Moons1_y))
print("Rand Index Value for Moons2: ", rand_index(y_pred4, Moons2_y))
print("Rand Index Value for Circles1: ", rand_index(y_pred5, Circles1_y))
print("Rand Index Value for Circles2: ", rand_index(y_pred6, Circles2_y))
```

```
Rand Index Value for Blobs1:  0.99911140760507
Rand Index Value for Blobs2:  0.9199715365799421
```

```
Rand Index Value for Moons1:  0.6201236379808761
Rand Index Value for Moons2:  0.6240836112964199
Rand Index Value for Circles1:  0.4996744496330887
Rand Index Value for Circles2:  0.4996744496330887
```

Answer: Decreasing order of Rand Index: 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: The rankins are same as K-means works well on globular data than the non-globular data.

1.0.3 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 will work well for Blobs1, Circles1 and Moons1 data. We know agglomerative clustering follows top-down approach and it tries to connect two closest pairs of points for different clusters.

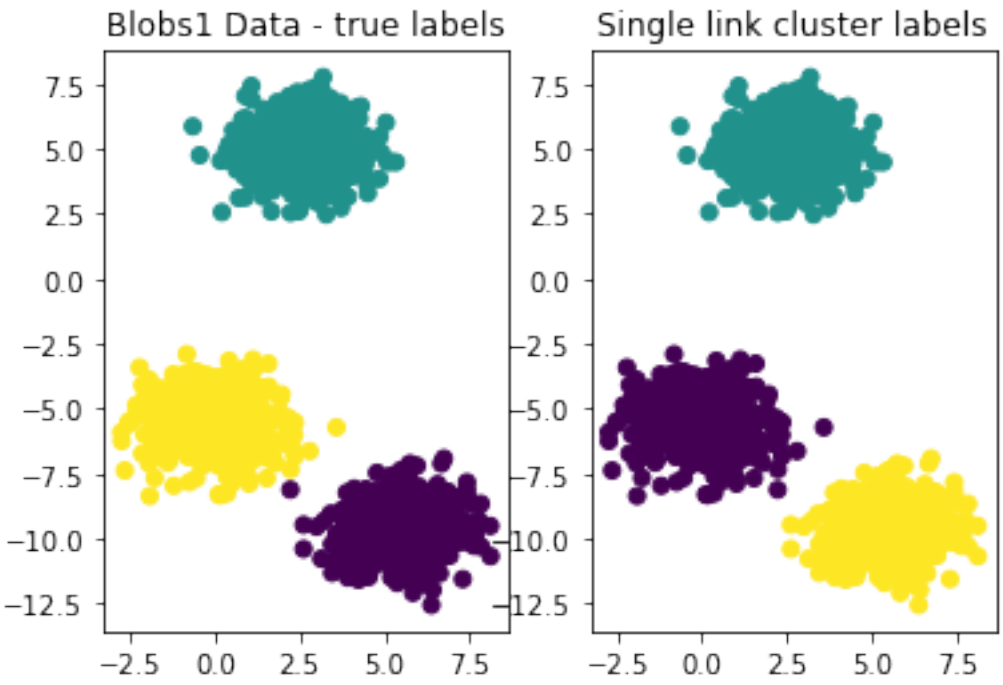
Question 2b: Without running Single-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Single-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

Answer: As the datapoints in Blobs2, Circles2 and Moons2 are very close to each other, hence the single link will not perfom well over these datasets. Single Link tries to connect closest pairs of points, but in these datasets inter-cluster points are very close to each other.

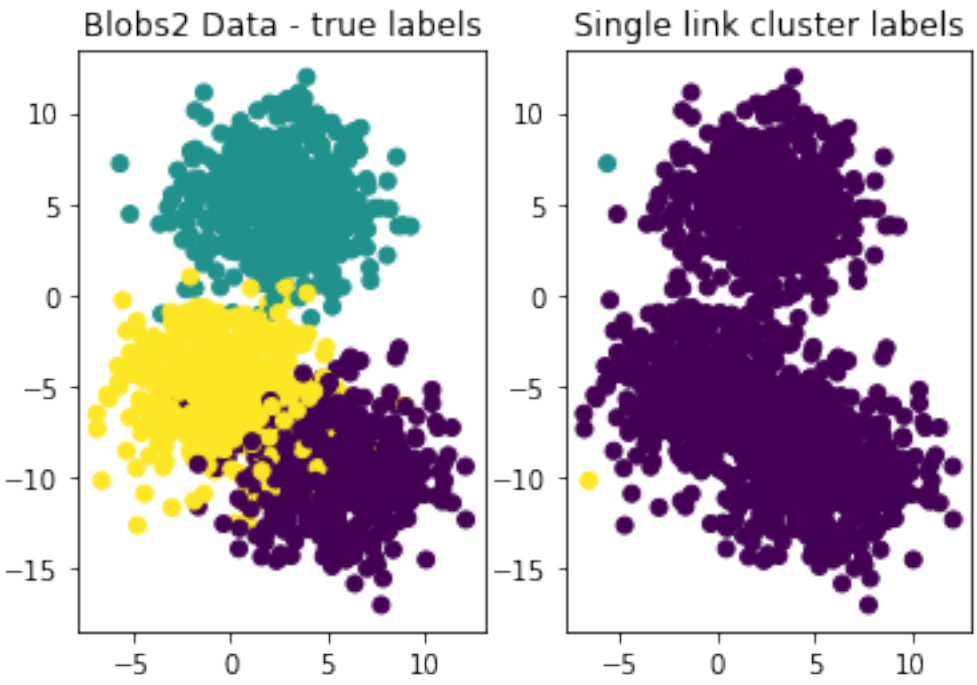
Question 2c: Run Single-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Single-link agglomerative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

[16]:
```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single",␣
 ↪n_clusters=n_clusters)
y_pred7 = single_linkage.fit_predict(Blobs1_X)
y_pred8 = single_linkage.fit_predict(Blobs2_X)
```

[17]:
```
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_pred7)
plt.title('Single link cluster labels')
plt.show()
```

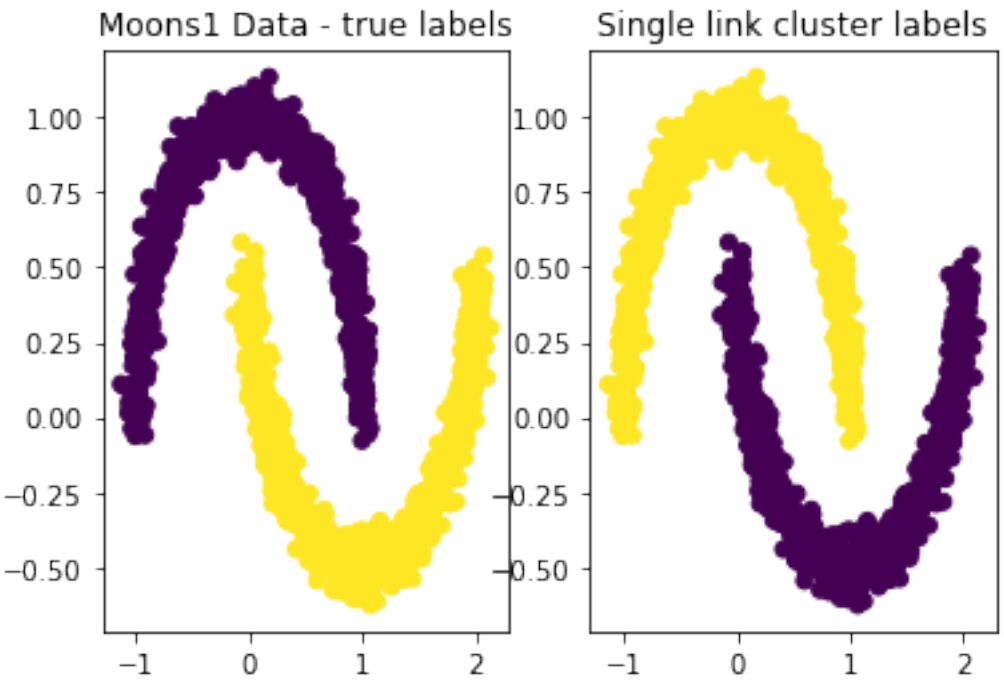


[18]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred8)
plt.title('Single link cluster labels')
plt.show()
```

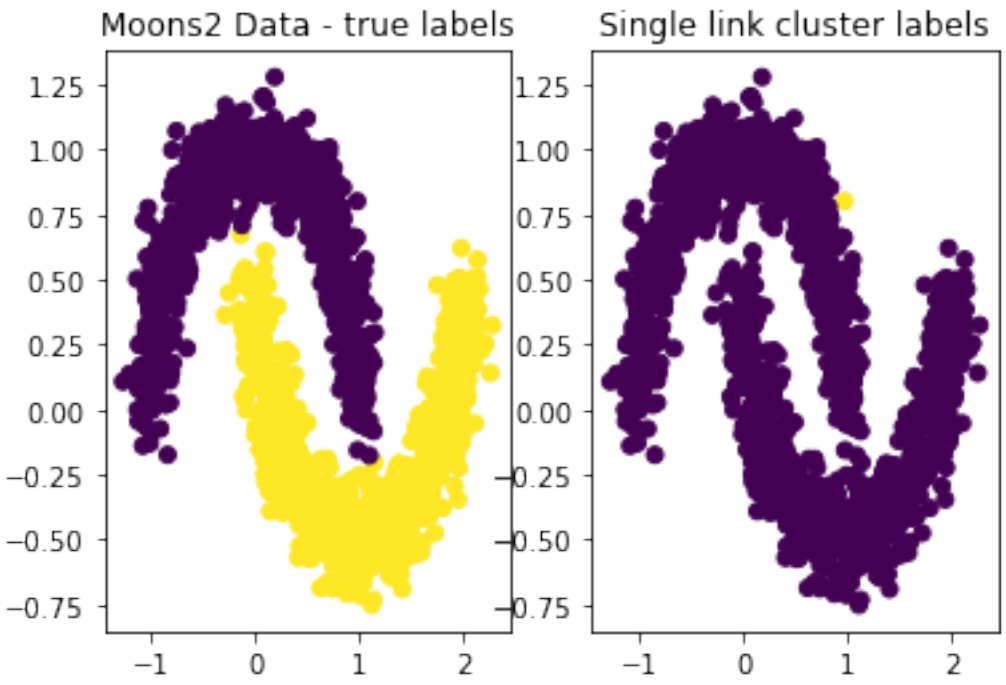


[19]:
```
n_clusters = 2
single_linkage = AgglomerativeClustering(linkage="single",␣
 ↪n_clusters=n_clusters)
y_pred9 = single_linkage.fit_predict(Moons1_X)
y_pred10 = single_linkage.fit_predict(Moons2_X)
y_pred11 = single_linkage.fit_predict(Circles1_X)
y_pred12 = single_linkage.fit_predict(Circles2_X)
```

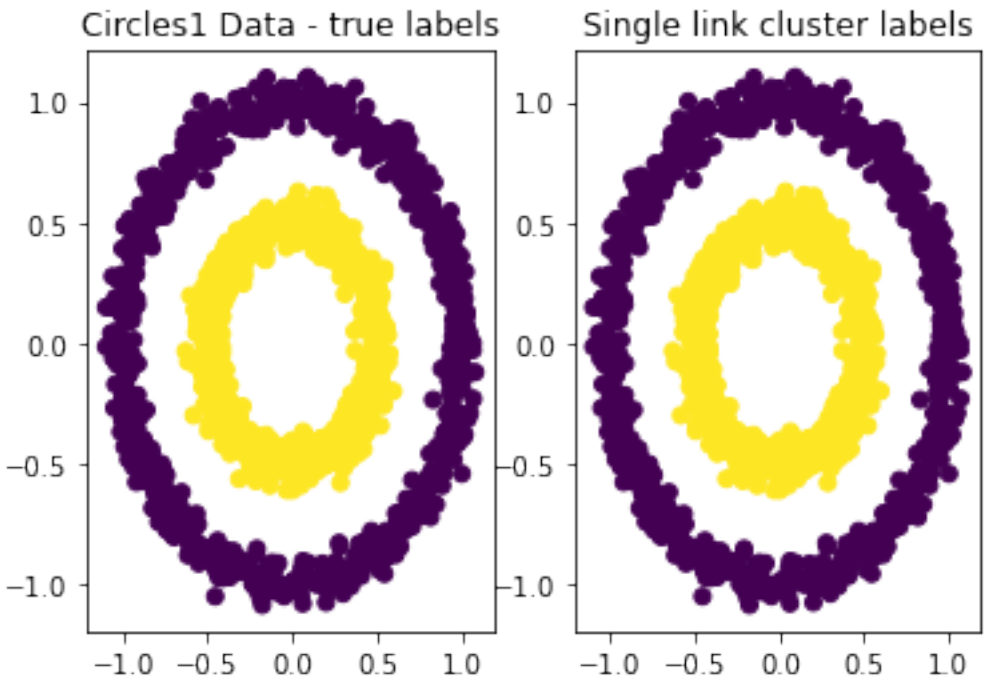
[20]:
```
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred9)
plt.title('Single link cluster labels')
plt.show()
```



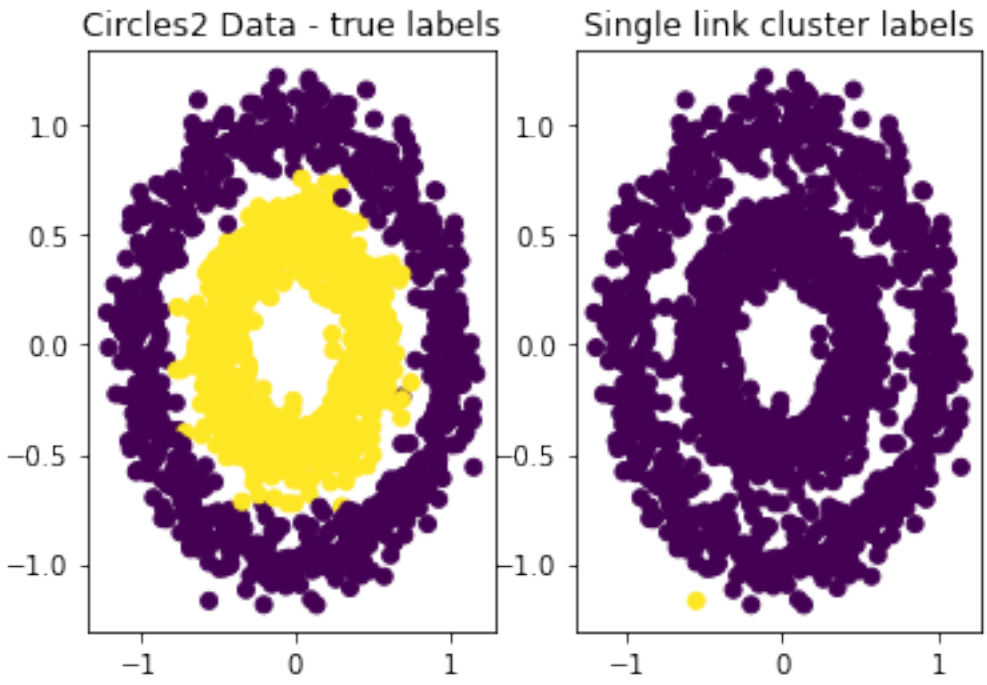
[21]:
```
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred10)
plt.title('Single link cluster labels')
plt.show()
```



[22]:
```
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_pred11)
plt.title('Single link cluster labels')
plt.show()
```



[23]:
```
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_pred12)
plt.title('Single link cluster labels')
plt.show()
```



Answer: Ranking in decreasing order based on visulized performance: Blobs1, Circles1, Moons1, Blobs2, Moons2, Circles2. Single link tries to find the closest pair of points between two clusters, so as the inter-cluster points in Blobs1, Moons1 and Circles1 are not that close enough, hence it will work well there.

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.

[24]:
```
print("Single Link Rand Index Value for Blobs1: ", rand_index(y_pred7,␣
 ↪Blobs1_y))
print("Single Link Rand Index Value for Blobs2: ", rand_index(y_pred8,␣
 ↪Blobs2_y))
print("Single Link Rand Index Value for Moons1: ", rand_index(y_pred9,␣
 ↪Moons1_y))
print("Single Link Rand Index Value for Moons2: ", rand_index(y_pred10,␣
 ↪Moons2_y))
print("Single Link Rand Index Value for Circles1: ", rand_index(y_pred11,␣
 ↪Circles1_y))
print("Single Link Rand Index Value for Circles2: ", rand_index(y_pred12,␣
 ↪Circles2_y))
```

```
Single Link Rand Index Value for Blobs1:  0.99911140760507
Single Link Rand Index Value for Blobs2:  0.33377896375361354
Single Link Rand Index Value for Moons1:  1.0
```

```
Single Link Rand Index Value for Moons2:  0.49966733377807426
Single Link Rand Index Value for Circles1:  1.0
Single Link Rand Index Value for Circles2:  0.49966733377807426
```

Answer: Decreasing order of Rand Index: 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.

Answer: Rankings are not in same order, but as single link tries to connect closest pair of points for different clusters so, it will work better for Moons1 and Circles1 than Blobs1, as the points are not that much randomized.

1.0.4 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: Complete link will work well on Blobs1, Circles1 and Moons1 data as it searches for highest pairwise distance between two clusters.

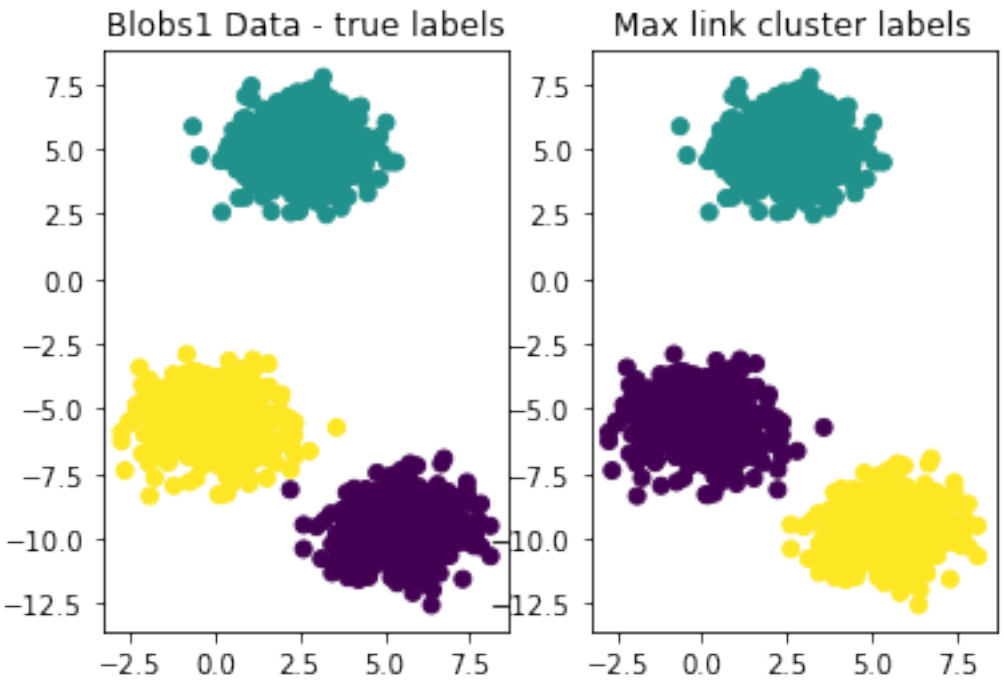
Question 3b: Without running Max-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Max-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

Answer: Max-link or complete link will not work well on Blobs2, Circles2 and Moons2 datasets. The data points in these datasets are very close to each other and complete link will try to look for longest distance between two clusters.

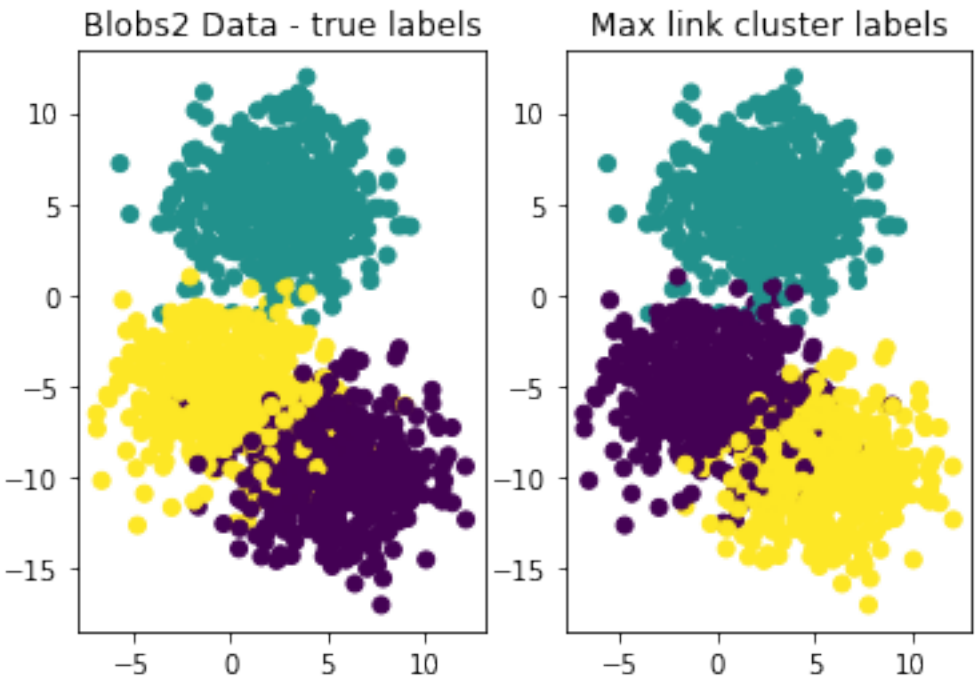
Question 3c: Run Max-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Max-link agglomerative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

[25]:
```
n_clusters = 3
complete_linkage = AgglomerativeClustering(linkage="complete",␣
 ↪n_clusters=n_clusters)
y_pred13 = complete_linkage.fit_predict(Blobs1_X)
y_pred14 = complete_linkage.fit_predict(Blobs2_X)
```

[26]:
```
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_pred13)
plt.title('Max link cluster labels')
plt.show()
```

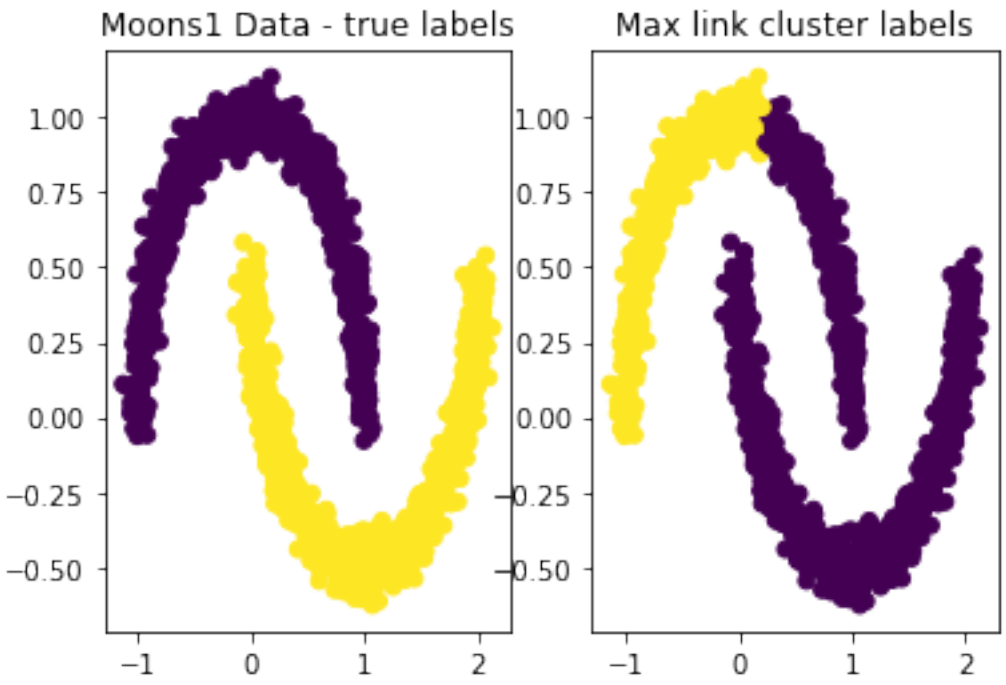


[27]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred13)
plt.title('Max link cluster labels')
plt.show()
```

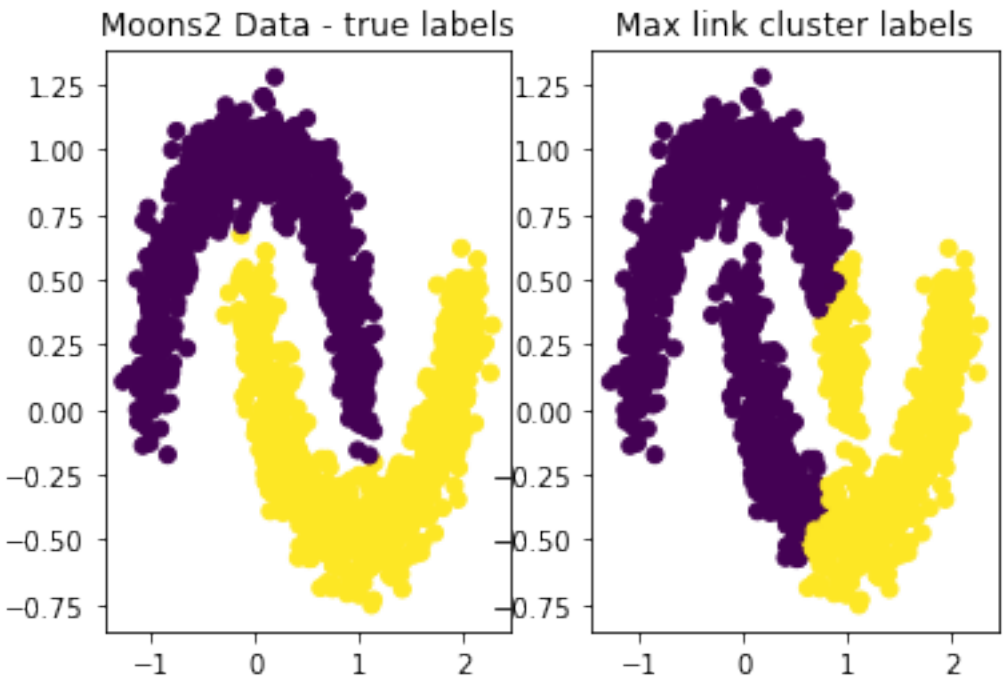


[28]:
```
n_clusters = 2
complete_linkage = AgglomerativeClustering(linkage="complete",␣
 ↪n_clusters=n_clusters)
y_pred15 = complete_linkage.fit_predict(Moons1_X)
y_pred16 = complete_linkage.fit_predict(Moons2_X)
y_pred17 = complete_linkage.fit_predict(Circles1_X)
y_pred18 = complete_linkage.fit_predict(Circles2_X)
```

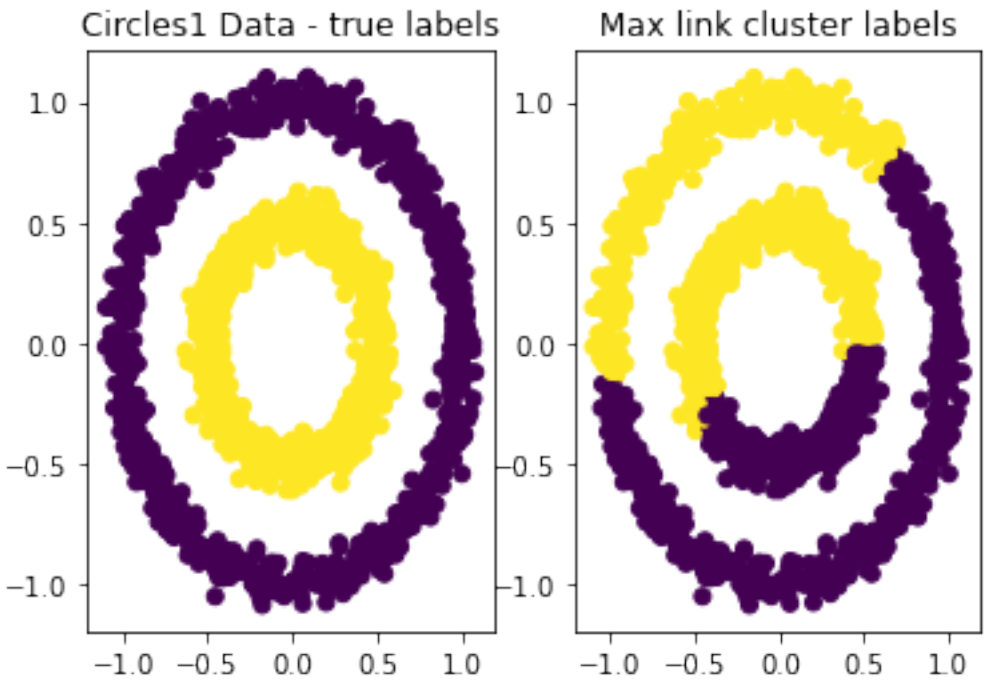
[29]:
```
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred15)
plt.title('Max link cluster labels')
plt.show()
```



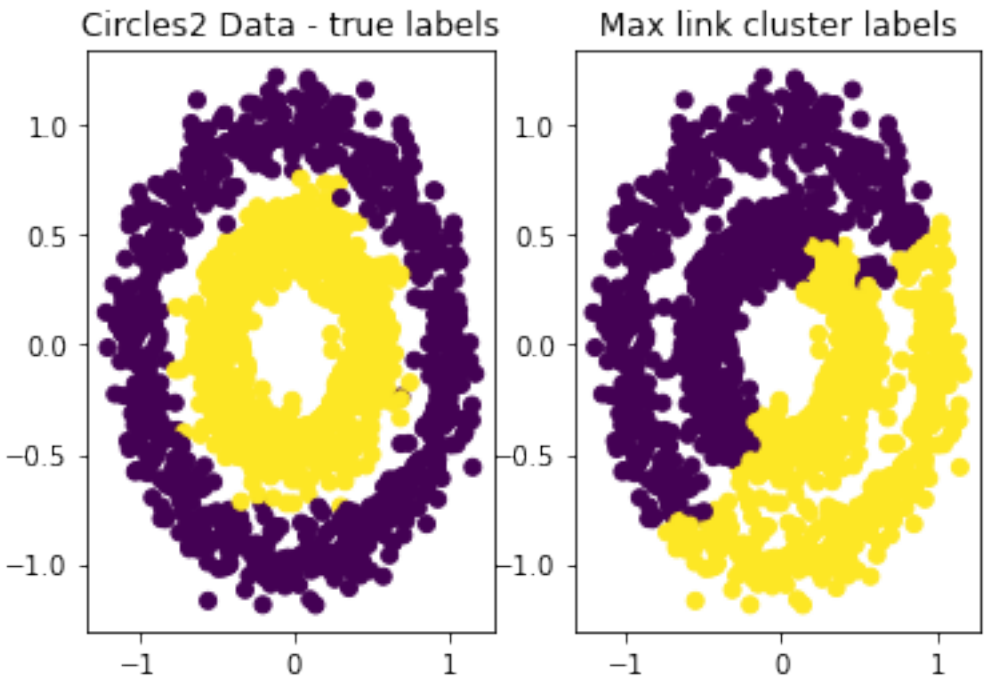
[30]:
```
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2 Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred16)
plt.title('Max link cluster labels')
plt.show()
```



[31]:
```
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_pred17)
plt.title('Max link cluster labels')
plt.show()
```



[32]:
```
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_pred18)
plt.title('Max link cluster labels')
plt.show()
```



Answer: Ranking in decreasing order based on visulized performance: Blobs1, Blobs2, Moons1, Circles1, Moons2, Circles2. This is because number of misclassified points are comparatively less than the actual output for all the datasets.

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.

[33]:
```
print("Max Link Rand Index Value for Blobs1: ", rand_index(y_pred13, Blobs1_y))
print("Max Link Rand Index Value for Blobs2: ", rand_index(y_pred14, Blobs2_y))
print("Max Link Rand Index Value for Moons1: ", rand_index(y_pred15, Moons1_y))
print("Max Link Rand Index Value for Moons2: ", rand_index(y_pred16, Moons2_y))
print("Max Link Rand Index Value for Circles1: ", rand_index(y_pred17,␣
 ↪Circles1_y))
print("Max Link Rand Index Value for Circles2: ", rand_index(y_pred18,␣
 ↪Circles2_y))
```

```
Max Link Rand Index Value for Blobs1:  0.99911140760507
Max Link Rand Index Value for Blobs2:  0.7736544362908606
Max Link Rand Index Value for Moons1:  0.662605292417167
Max Link Rand Index Value for Moons2:  0.5965310206804536
Max Link Rand Index Value for Circles1:  0.5218714698688014
Max Link Rand Index Value for Circles2:  0.5000587058038692
```

Answer: Decreasing order of Rand Index: 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.

Answer: No, it is not the same when we compare 3(c) and 3(d) because the number of points in circles dataset and moons dataset affect the rand index value. Number of points in circles1 is greater than number of points in moons2.

1.0.5 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: Average Link will work for Blobs1 as the cluster in the dataset are properly seperated without any overlapping. Also, average link is not a robust algorithm, so it may not predict the cluster as required.

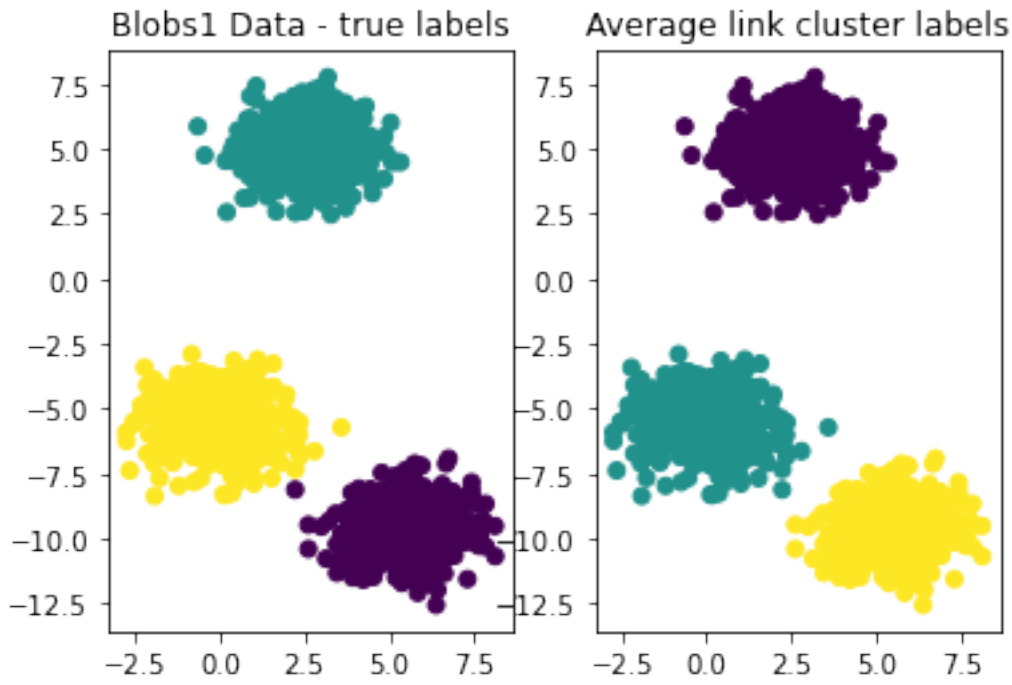
Question 4b: Without running Average-link agglomerative clustering, for all the datasets (except Rand) provided in the practice session, list the datasets where Average-link agglomerative clustering is expected to NOT work well. Support your answer by explaining your rationale.

Answer: Average link may not work well for Blobs2, Moons1, Moons2, Circles1, Circles2 because points in Blobs2 are overlapping so their mean will be affected. For Moons1, Moons2, Circles1 and Circles2 shape of the data affects the mean.

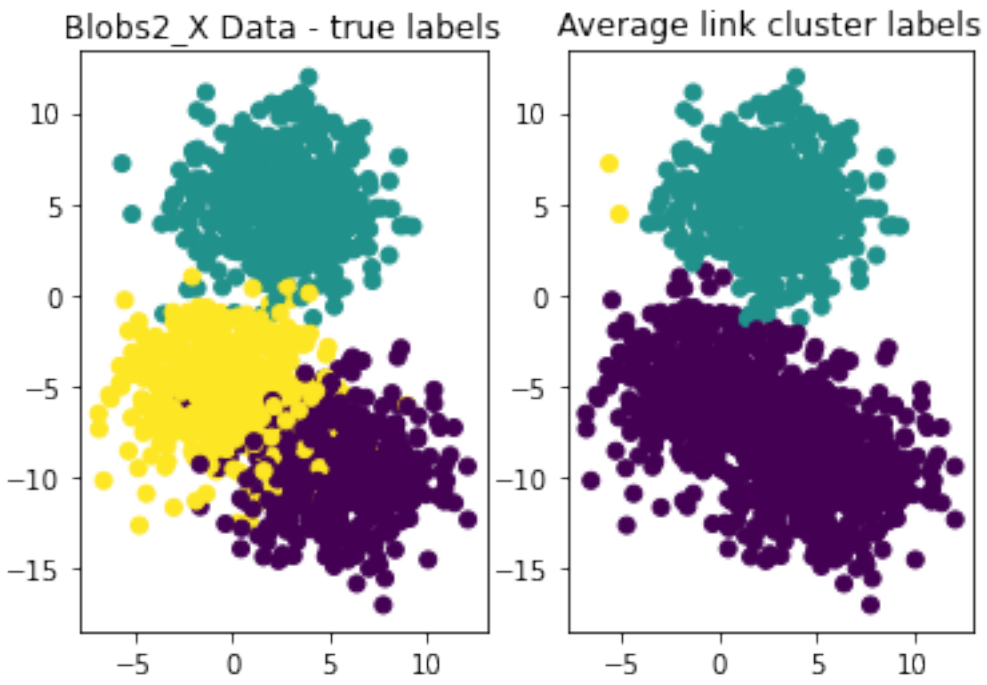
Question 4c: Run Average-link agglomerative clustering algorithm on all the datasets (except Rand). Choose n_clusters based on the number of clusters present in these datasets. Visualize the clusters for each of them. Based on the visualization, rank the datasets in decreasing order of Average-link agglomerative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

[34]:
```
n_clusters = 3
average_linkage = AgglomerativeClustering(linkage="average",␣
 ↪n_clusters=n_clusters)
y_pred19 = average_linkage.fit_predict(Blobs1_X)
y_pred20 = average_linkage.fit_predict(Blobs2_X)
```

[35]:
```
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_pred19)
plt.title('Average link cluster labels')
plt.show()
```

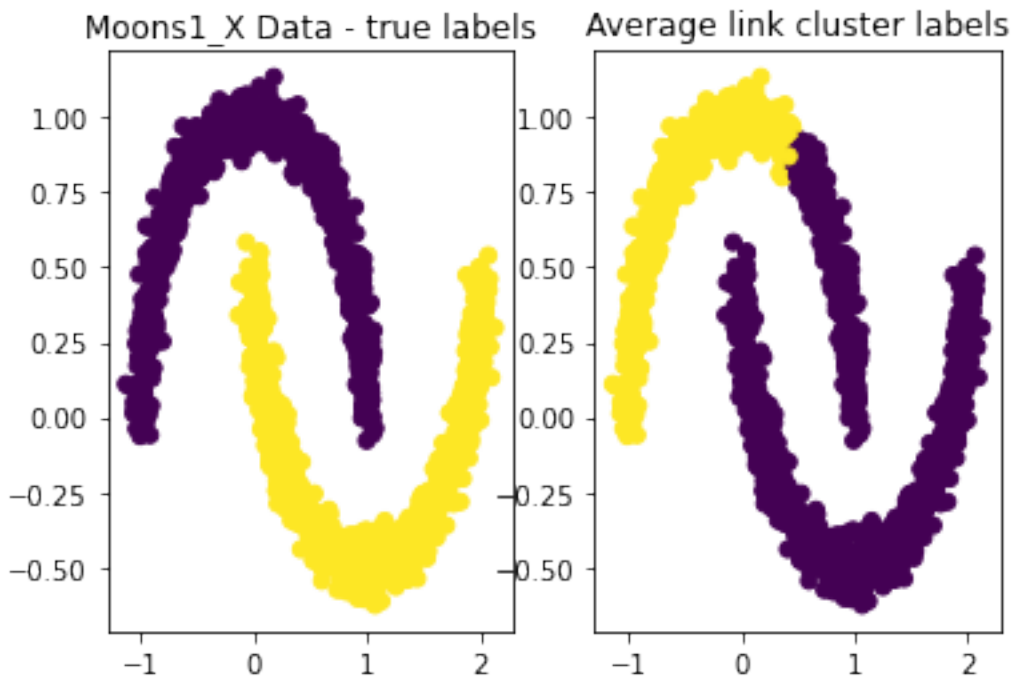


[36]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred20)
plt.title('Average link cluster labels')
plt.show()
```

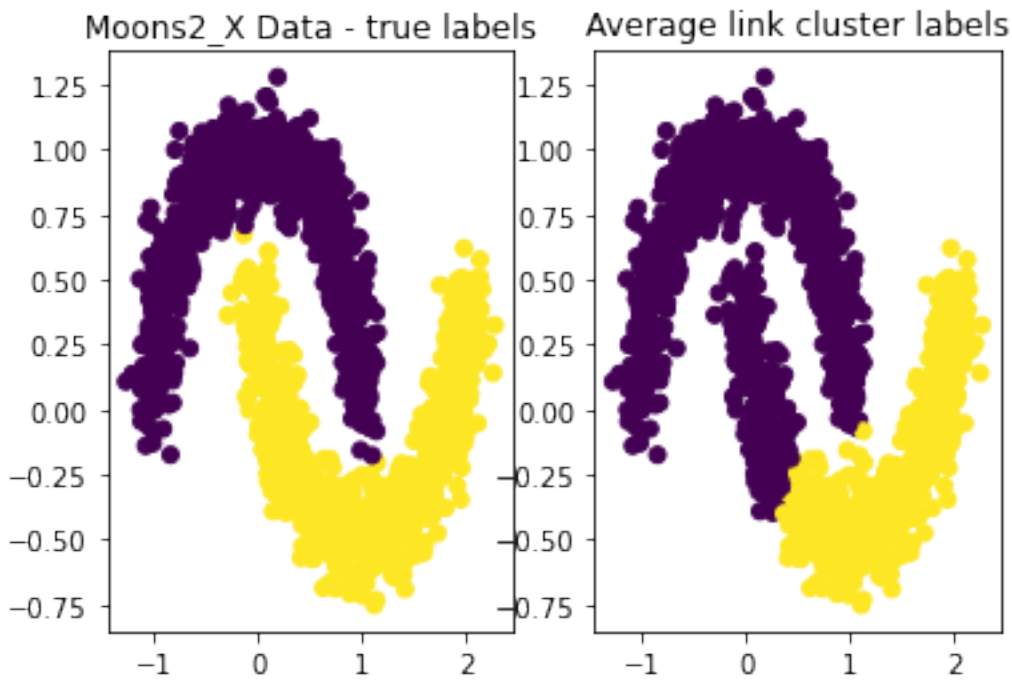


[37]:
```
n_clusters = 2
average_linkage = AgglomerativeClustering(linkage="average",␣
 ↪n_clusters=n_clusters)
y_pred21 = average_linkage.fit_predict(Moons1_X)
y_pred22 = average_linkage.fit_predict(Moons2_X)
y_pred23 = average_linkage.fit_predict(Circles1_X)
y_pred24 = average_linkage.fit_predict(Circles2_X)
```

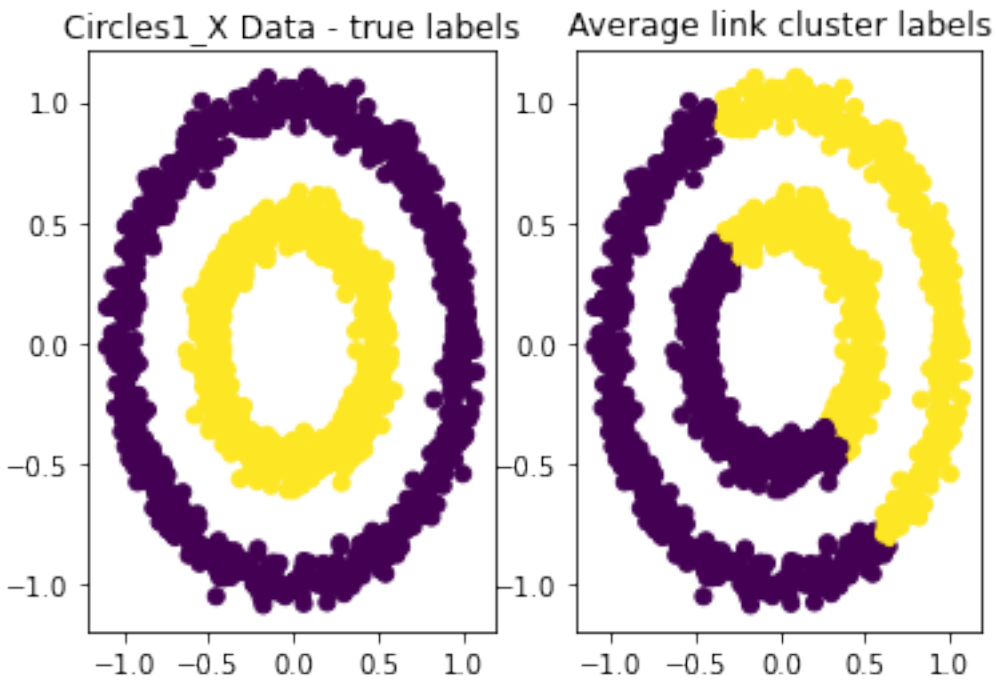
[38]:
```
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1_X Data - true labels')
plt.subplot(1,2,2)

plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred21)
plt.title('Average link cluster labels')
plt.show()
```



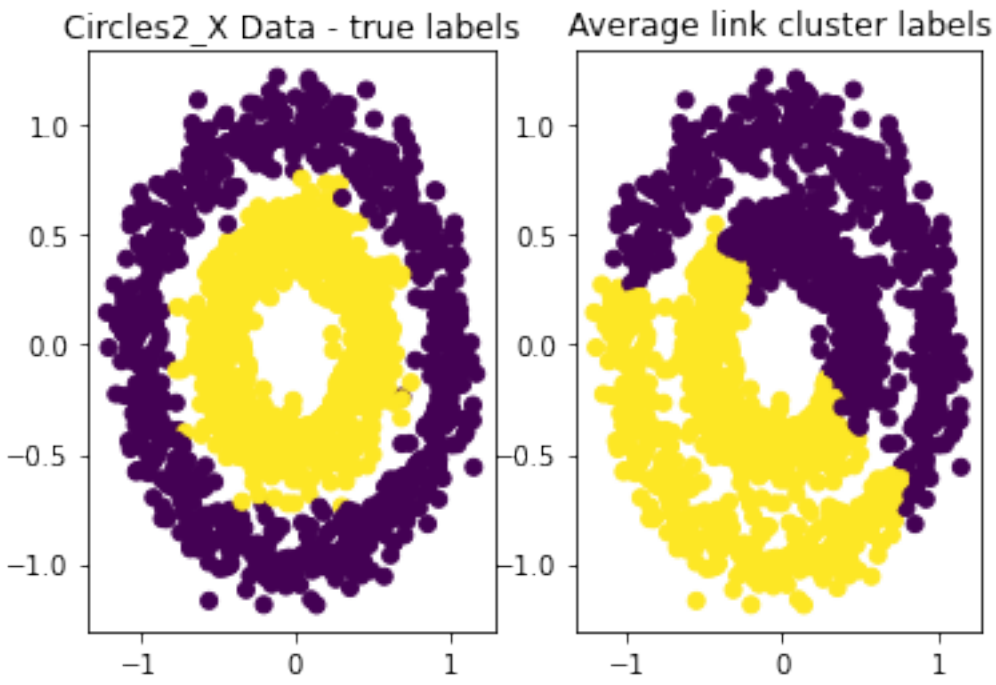
[39]:
```
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred22)
plt.title('Average link cluster labels')
plt.show()
```



[40]:
```
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred23)
plt.title('Average link cluster labels')
plt.show()
```



[41]:
```
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred24)
plt.title('Average link cluster labels')
plt.show()
```



Answer: Ranking in decreasing order based on visualization: Blobs1, Blobs2, Moons1, Moons2, Circles1, Circles2. Because the shape of the data affects the mean and data is well seperated and not overlapping in Blobs1, so it is ranked at the highest order than others.

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.

[42]:
```
print("Average Link Rand Index Value for Blobs1: ", rand_index(y_pred19,␣
 ↪Blobs1_y))
print("Average Link Rand Index Value for Blobs2: ", rand_index(y_pred20,␣
 ↪Blobs2_y))
print("Average Link Rand Index Value for Moons1: ", rand_index(y_pred21,␣
 ↪Moons1_y))
print("Average Link Rand Index Value for Moons2: ", rand_index(y_pred22,␣
 ↪Moons2_y))
print("Average Link Rand Index Value for Circles1: ", rand_index(y_pred23,␣
 ↪Circles1_y))
print("Average Link Rand Index Value for Circles2: ", rand_index(y_pred24,␣
 ↪Circles2_y))
```

```
Average Link Rand Index Value for Blobs1:  0.99911140760507
Average Link Rand Index Value for Blobs2:  0.7636575494774294
Average Link Rand Index Value for Moons1:  0.7132310429175005
Average Link Rand Index Value for Moons2:  0.7457647320435846
```

```
Average Link Rand Index Value for Circles1:  0.500414498554592
Average Link Rand Index Value for Circles2:  0.5050780520346898
```

Answer: Decreasing order of Rand Index: Blobs1, Blobs2, Moons2, Moons1, Circles2, Circles1

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: No, the ranking is not same when compared with question 4(c) and 4(d), because Moons1 and Circles1 have different shape and density when compared with Moons2 and Circles2.

1.0.6 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: DBSCAN algorithm basically depends on the density of the points in particular cluster which is determined using the epsilon value and minimum number of required points. So, if there is a dataset which is having clusters very close to each, or border points of clusters close to each other, then there DBSCAN will not perform properly. Depending on the value of epsilon and minpts, DBSCAN will work well for Blobs1, Moons1, Circles1 datasets as they are well seperated from the points of other clusters.

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.

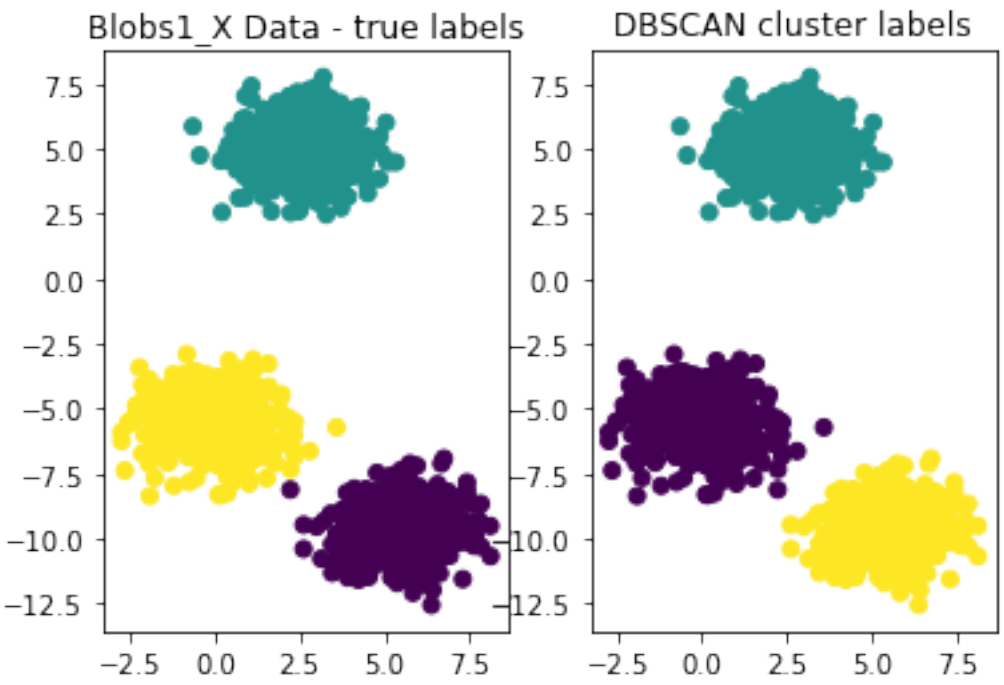
Answer: As discussed above DBSCAN depends on density of poitns for a particular cluster, hence it may not work well for datasets Blobs2, Moons2 and Circles2 as the points in different clusters are very close to each other.

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.

[43]:
```
dbscan = DBSCAN(eps=1.4, min_samples=10)
y_pred25 = dbscan.fit_predict(Blobs1_X)
np.sum(y_pred25==-1)
```

[43]: 0

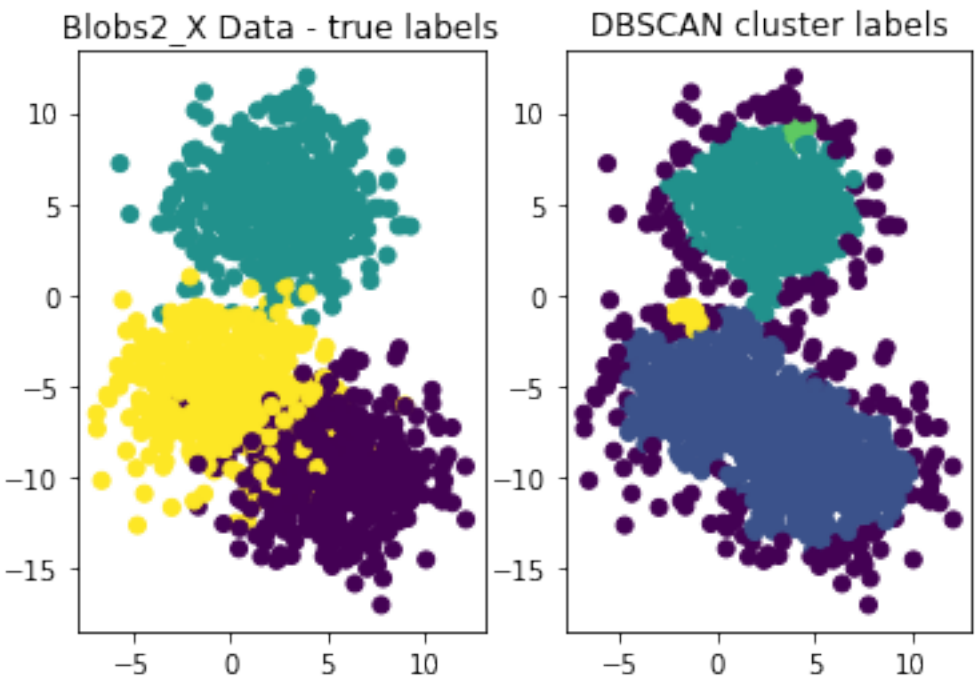
[44]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred25)
plt.title('DBSCAN cluster labels')
plt.show()
```



[45]:
```
dbscan = DBSCAN(eps=0.85, min_samples=10)
y_pred26 = dbscan.fit_predict(Blobs2_X)
np.sum(y_pred26==-1)
```

[45]: 218

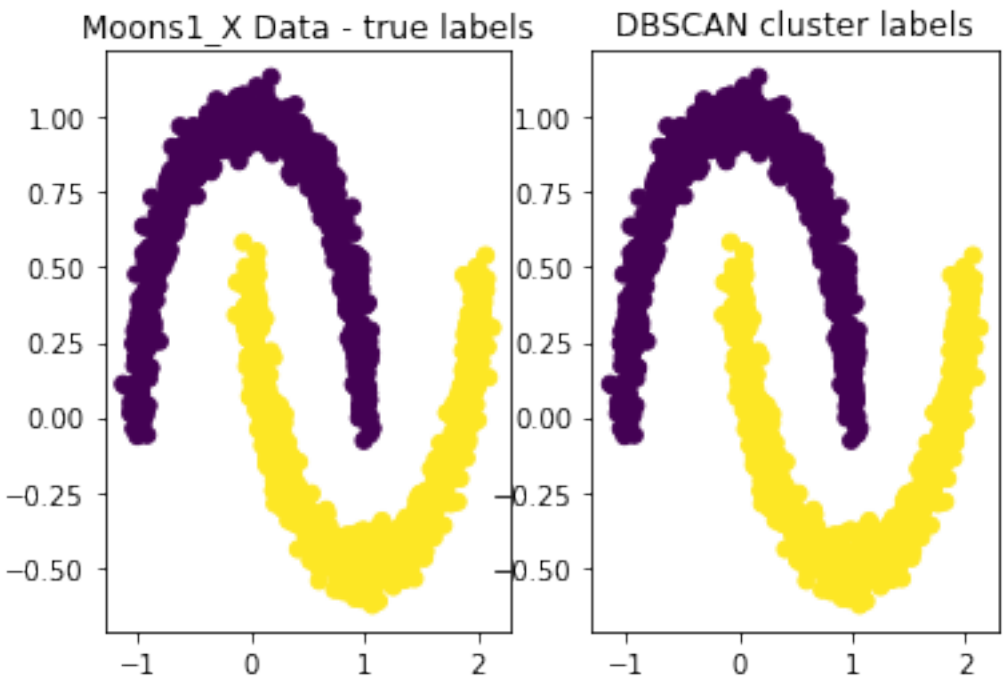
[46]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred26)
plt.title('DBSCAN cluster labels')
plt.show()
```



[47]:
```
dbscan = DBSCAN(eps=0.1, min_samples=10)
y_pred27 = dbscan.fit_predict(Moons1_X)
np.sum(y_pred27==-1)
```

[47]: 0

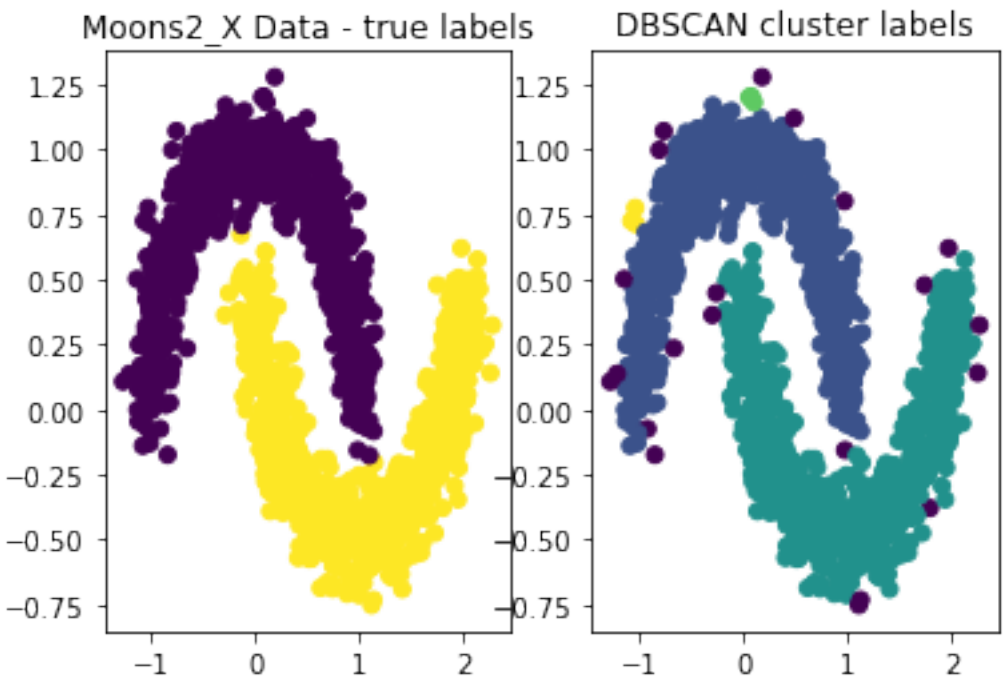
[48]:
```
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred27)
plt.title('DBSCAN cluster labels')
plt.show()
```



[49]:
```
dbscan = DBSCAN(eps=0.09, min_samples=3)
y_pred28 = dbscan.fit_predict(Moons2_X)
np.sum(y_pred28==-1)
```

[49]: 21

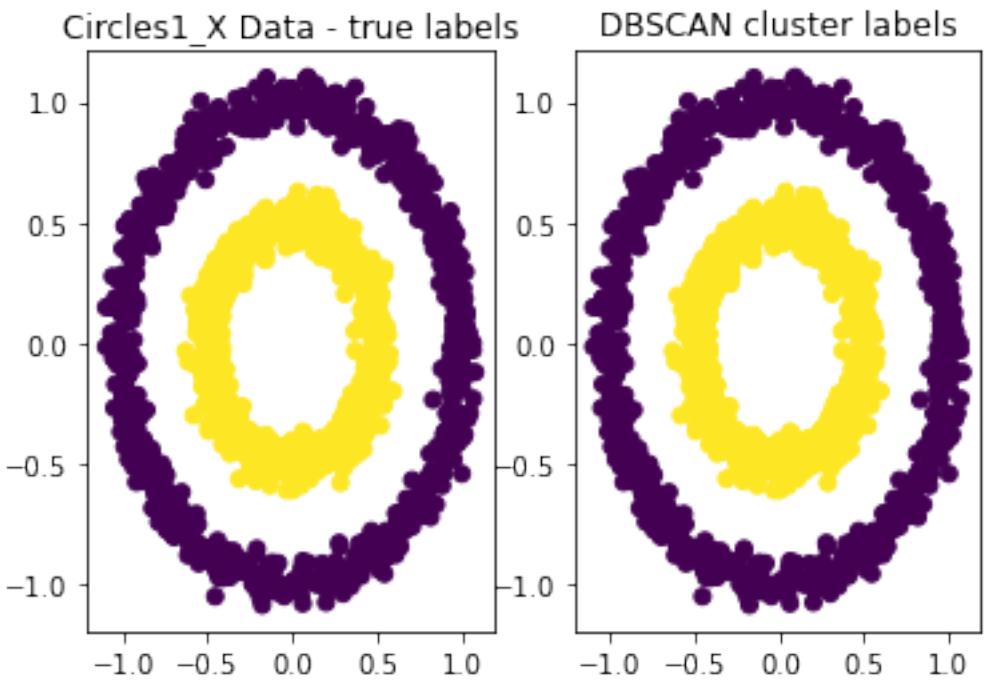
[50]:
```
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred28)
plt.title('DBSCAN cluster labels')
plt.show()
```



[51]:
```
dbscan = DBSCAN(eps=0.1, min_samples=10)
y_pred29 = dbscan.fit_predict(Circles1_X)
np.sum(y_pred29==-1)
```

[51]: 0

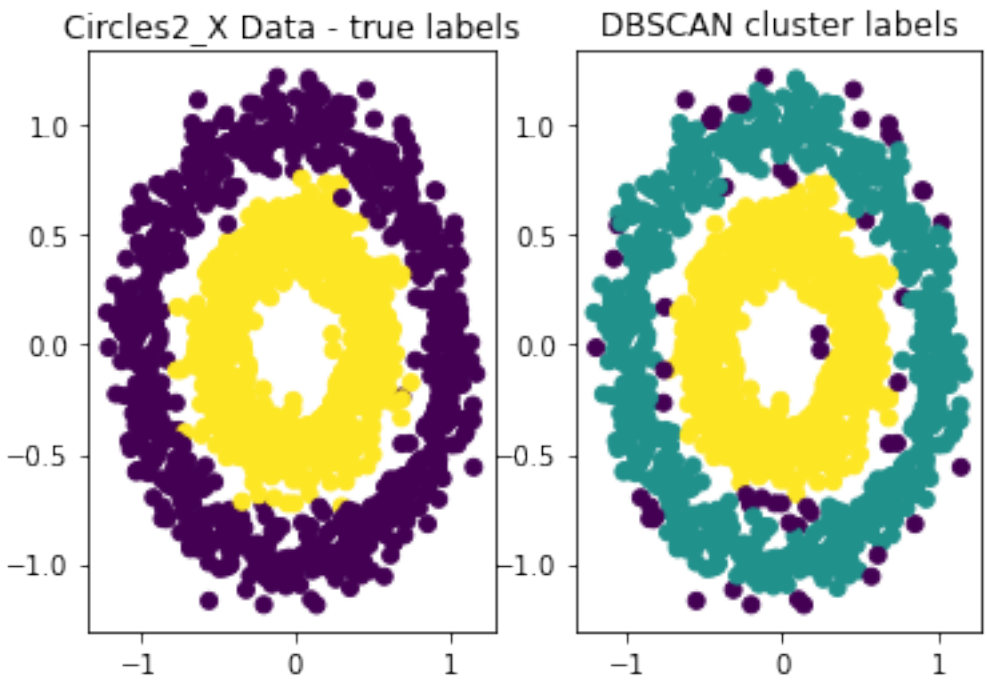
[52]:
```
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred29)
plt.title('DBSCAN cluster labels')
plt.show()
```



[53]:
```
dbscan = DBSCAN(eps=0.09, min_samples=6)
y_pred30 = dbscan.fit_predict(Circles2_X)
np.sum(y_pred30==-1)
```

[53]: 54

[54]:
```
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred30)
plt.title('DBSCAN cluster labels')
plt.show()
```



Answer: Decreasing order of performance based on visualization: Blobs1, Moons1, Circles1, Moons2, Circles2, Blobs2. Based on the chosen epsilon and minimum points for all the data sets it can be noticed that datasets with less affinity between two clusters tends to not form the actual clusters. The clusters for Circles2, Moons2 and Blobs2 have multiple noise points and also they are not same as the actual outputs. Hence, is the order of performance.

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?

Answer: Blobs1, Moons1 and Circles1 have least number of noise points as 0, because these clusters are well seperated from others and also their internal densities is very high, so they are able to connect maximal core points to form the clusters same as actual outputs. Also, Blobs2 has 218 noise points, which can be the best possible outcome. Moons2 has 21 noise points and Circles2 has 54 noise 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.

[55]:
```
print("DBSCAN Rand Index Value for Blobs1: ", rand_index(y_pred25, Blobs1_y))
print("DBSCAN Rand Index Value for Blobs2: ", rand_index(abs(y_pred26),␣
↪Blobs2_y))
print("DBSCAN Rand Index Value for Moons1: ", rand_index(y_pred27, Moons1_y))
print("DBSCAN Rand Index Value for Moons2: ", rand_index(abs(y_pred28),␣
↪Moons2_y))
```

```
print("DBSCAN Rand Index Value for Circles1: ", rand_index(y_pred29,␣
 ↪Circles1_y))
print("DBSCAN Rand Index Value for Circles2: ", rand_index(abs(y_pred30),␣
 ↪Circles2_y))
```

```
DBSCAN Rand Index Value for Blobs1:  0.99911140760507
DBSCAN Rand Index Value for Blobs2:  0.7131670002223705
DBSCAN Rand Index Value for Moons1:  1.0
DBSCAN Rand Index Value for Moons2:  0.9775912830776073
DBSCAN Rand Index Value for Circles1:  1.0
DBSCAN Rand Index Value for Circles2:  0.9355125639315099
```

Answer: Decreasing order of Rand Index: 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.

Answer: No, the rankings in 5(c) are not in consideration with rankings in 5(d) because Blobs1 dataset is having 3 points which were not clustered correctly as per the actual output, as they are not close to minpts of same cluster in actual output.

1.0.7 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: If we consider the affinity matrix to be created based on the mutual KNN, then it will work well for Blobs1, Blob2

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.

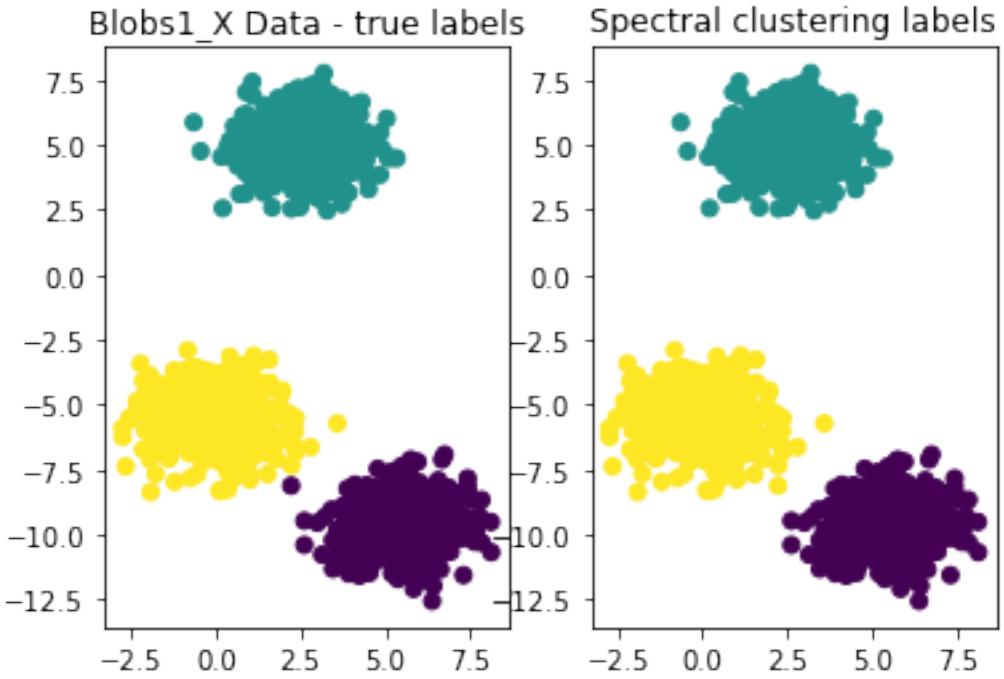
Answer: It may not work well for Moons1, Moons2, Circles1 and Circles2 as the affinity is considered to be mutual KNN.

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.

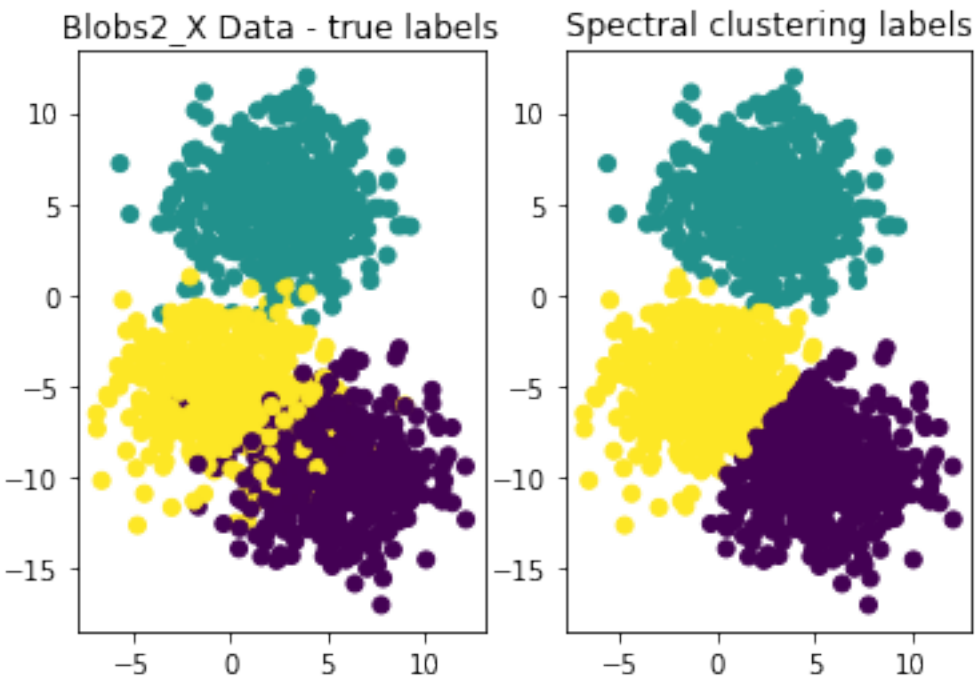
[56]:
```
n_clusters = 3
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state)
y_pred31 = spectral.fit_predict(Blobs1_X)
y_pred32 = spectral.fit_predict(Blobs2_X)
```

[57]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=Blobs1_y)
plt.title('Blobs1_X Data - true labels')
plt.subplot(1,2,2)
```

```
plt.scatter(Blobs1_X[:, 0], Blobs1_X[:, 1], c=y_pred31)
plt.title('Spectral clustering labels')
plt.show()
```

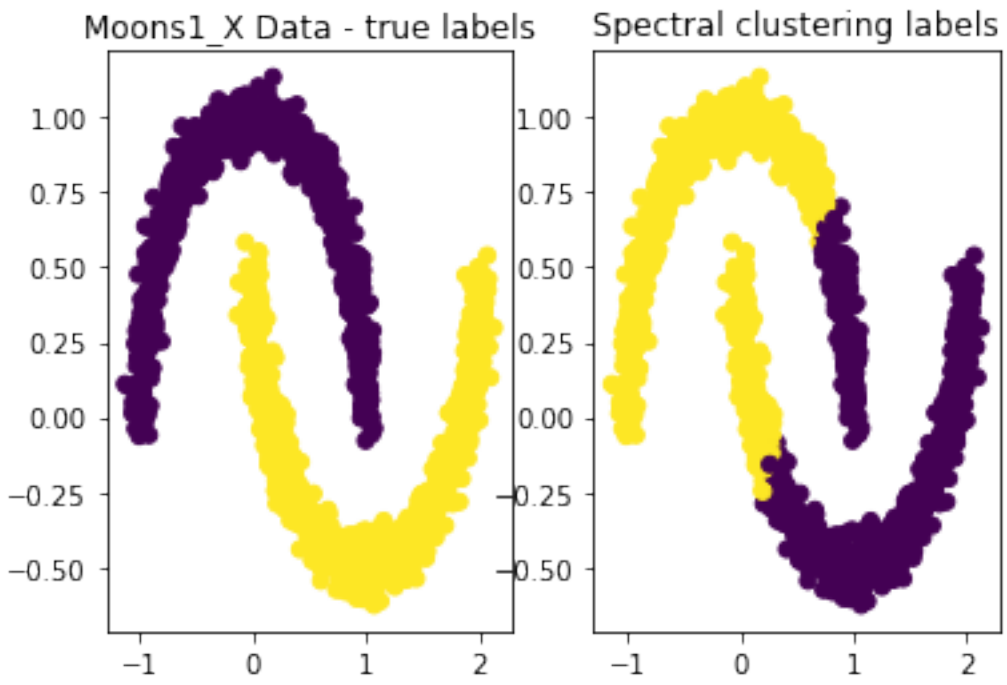


[58]:
```
plt.subplot(1,2,1)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=Blobs2_y)
plt.title('Blobs2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Blobs2_X[:, 0], Blobs2_X[:, 1], c=y_pred32)
plt.title('Spectral clustering labels')
plt.show()
```

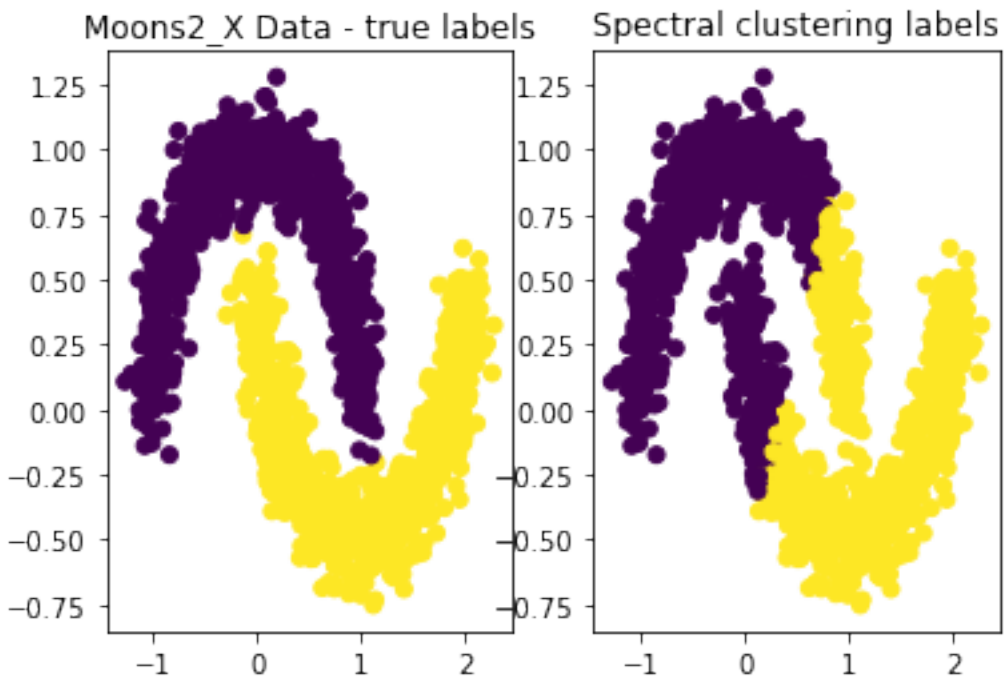


[59]:
```
n_clusters = 2
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state)
y_pred33 = spectral.fit_predict(Moons1_X)
y_pred34 = spectral.fit_predict(Moons2_X)
y_pred35 = spectral.fit_predict(Circles1_X)
y_pred36 = spectral.fit_predict(Circles2_X)
```

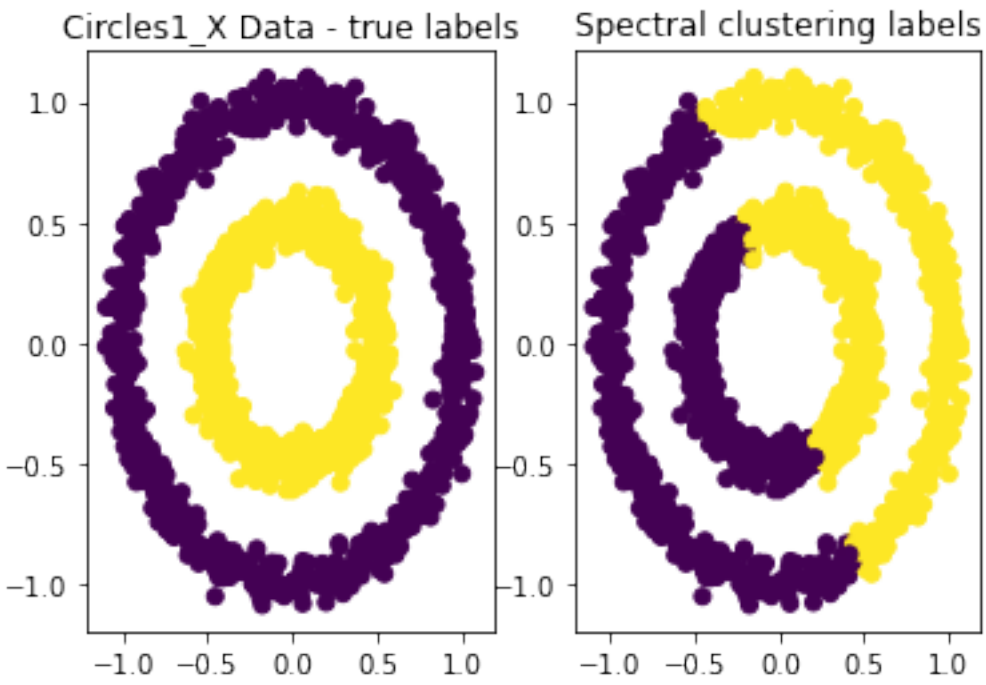
[60]:
```
plt.subplot(1,2,1)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=Moons1_y)
plt.title('Moons1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons1_X[:, 0], Moons1_X[:, 1], c=y_pred33)
plt.title('Spectral clustering labels')
plt.show()
```



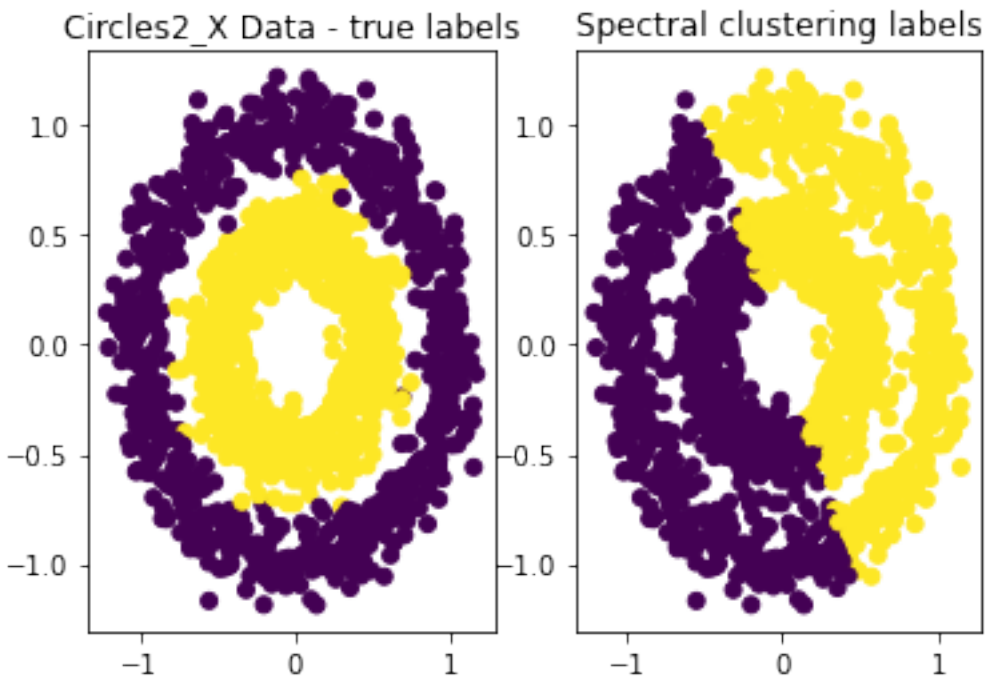
[61]:
```
plt.subplot(1,2,1)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=Moons2_y)
plt.title('Moons2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Moons2_X[:, 0], Moons2_X[:, 1], c=y_pred34)
plt.title('Spectral clustering labels')
plt.show()
```



[62]:
```
plt.subplot(1,2,1)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=Circles1_y)
plt.title('Circles1_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles1_X[:, 0], Circles1_X[:, 1], c=y_pred35)
plt.title('Spectral clustering labels')
plt.show()
```



[63]:
```
plt.subplot(1,2,1)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=Circles2_y)
plt.title('Circles2_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Circles2_X[:, 0], Circles2_X[:, 1], c=y_pred36)
plt.title('Spectral clustering labels')
plt.show()
```



Answer: Ranking in decreasing order of Blobs1, Blobs2, Moons2, Moons1, Circles2 and Circles1. It depends on the affinity measure considered to do clustering. If we consider mutual KNN, it works well for Blobs1 and Blosb2.

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.

[64]:
```
print("Spectral Clustering Rand Index Value for Blobs1: ", rand_index(y_pred31,␣
 ↪Blobs1_y))
print("Spectral Clustering Rand Index Value for Blobs2: ", rand_index(y_pred32,␣
 ↪Blobs2_y))
print("Spectral Clustering Rand Index Value for Moons1: ", rand_index(y_pred33,␣
 ↪Moons1_y))
print("Spectral Clustering Rand Index Value for Moons2: ", rand_index(y_pred34,␣
 ↪Moons2_y))
print("Spectral Clustering Rand Index Value for Circles1: ",␣
 ↪rand_index(y_pred35, Circles1_y))
print("Spectral Clustering Rand Index Value for Circles2: ",␣
 ↪rand_index(y_pred36, Circles2_y))
```

```
Spectral Clustering Rand Index Value for Blobs1:  0.99911140760507
Spectral Clustering Rand Index Value for Blobs2:  0.919189682010229
Spectral Clustering Rand Index Value for Moons1:  0.6441263064265066
Spectral Clustering Rand Index Value for Moons2:  0.6448441183010896
```

```
Spectral Clustering Rand Index Value for Circles1:  0.49966733377807426
Spectral Clustering Rand Index Value for Circles2:  0.4997553924838781
```

Answer: Decreasing order of Rand Index: Blobs1, Blobs2, Moons2, Moons1, Circles2, Circles1

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, they are in consideration as affinity measure works well for Blobs1 and Blobs2 than any other datasets.

1.0.8 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: Clustering tendency say us if the data has a non-random structure or not. It helps us determin correct number of clusters, helps in evaluation of cluster results. All this can be done perfectly for datasets such as Blobs1, Moons1 and Circles1. It can also be evaluated for Blobs2, Moons2 and Circles2, but their clustering tendency will not be as good as other three datasets.

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 dataset cannot give us good clustering tendency because it has random structure which doesn't helps us to define any new clusters.

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

[65]:
```
hopkins(Blobs1_X)
```

[65]:
```
0.9406464852873702
```

[66]:
```
hopkins(Blobs2_X)
```

[66]:
```
0.8158832326487225
```

[67]:
```
hopkins(Moons1_X)
```

[67]:
```
0.9280107298653895
```

[68]:
```
hopkins(Moons2_X)
```

[68]:
```
0.8670139903310856
```

[69]:
```
hopkins(Circles1_X)
```

[69]:
```
0.8421611174913359
```

[70]:
```
hopkins(Circles2_X)
```

[70]:
```
0.7662094824789657
```

[71]:
```
hopkins(Rand_X)
```

[71]:
```
0.623559513097174
```

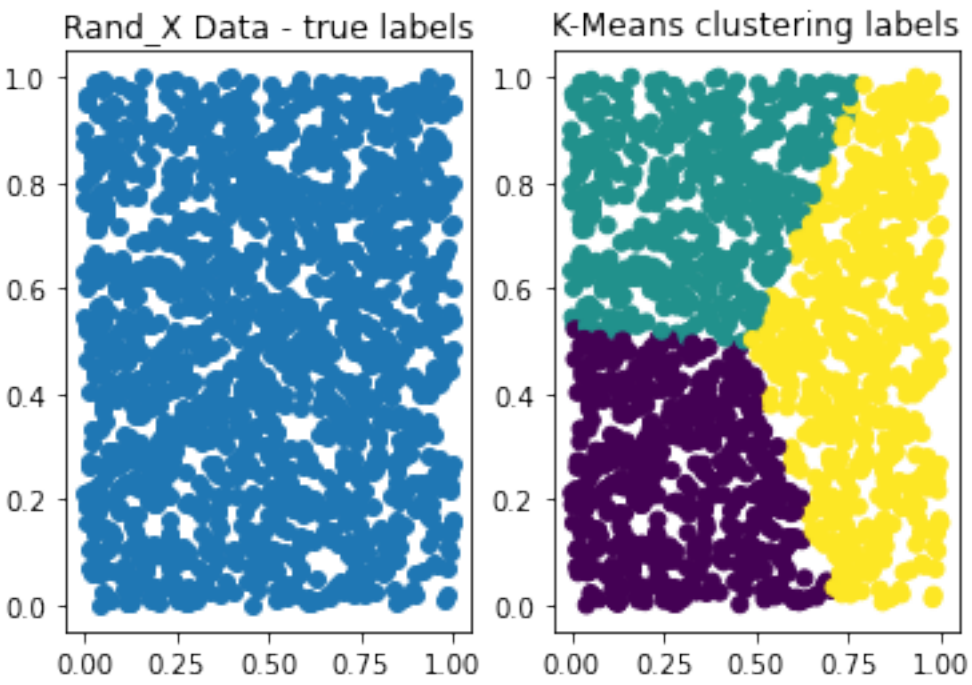
Answer: Decreasing order of Hopkins Statistics for all datasets: Blobs1, Moons1, Moons2, Blobs2, Circles1, Circles2, Rand.

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

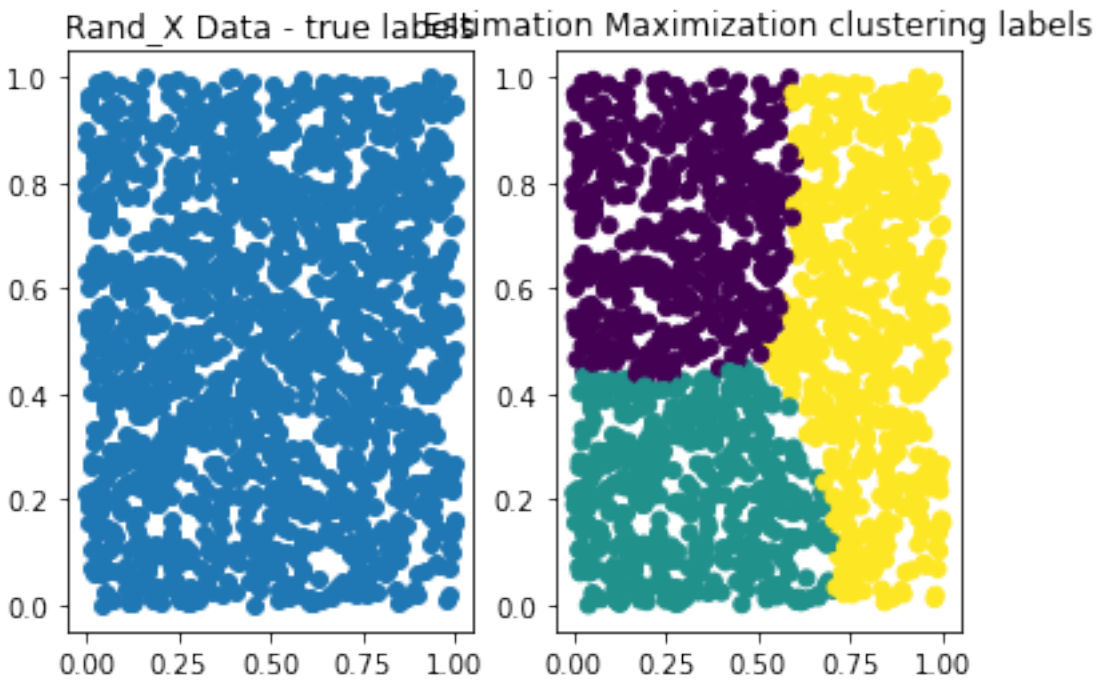
Answer: No, they are not in consideration when compared for 7(a) and 7(b) because the data in Blobs1, Moons1, Moons2 is highly clustered and regularly distributed.

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.

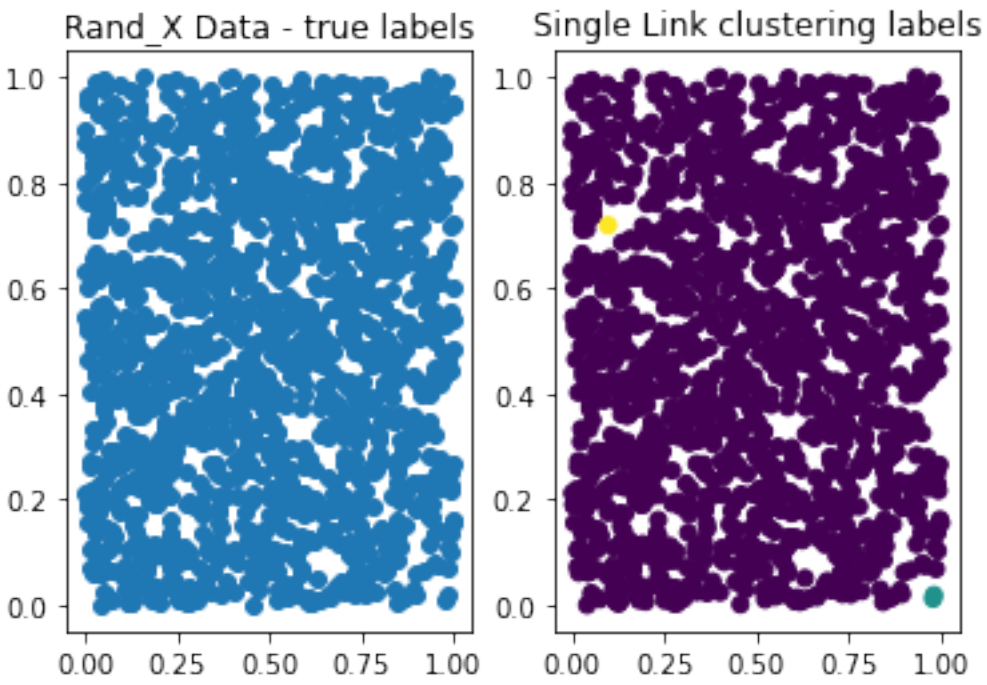
[72]:
```
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred37 = kmeans.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred37)
plt.title('K-Means clustering labels')
plt.show()
```



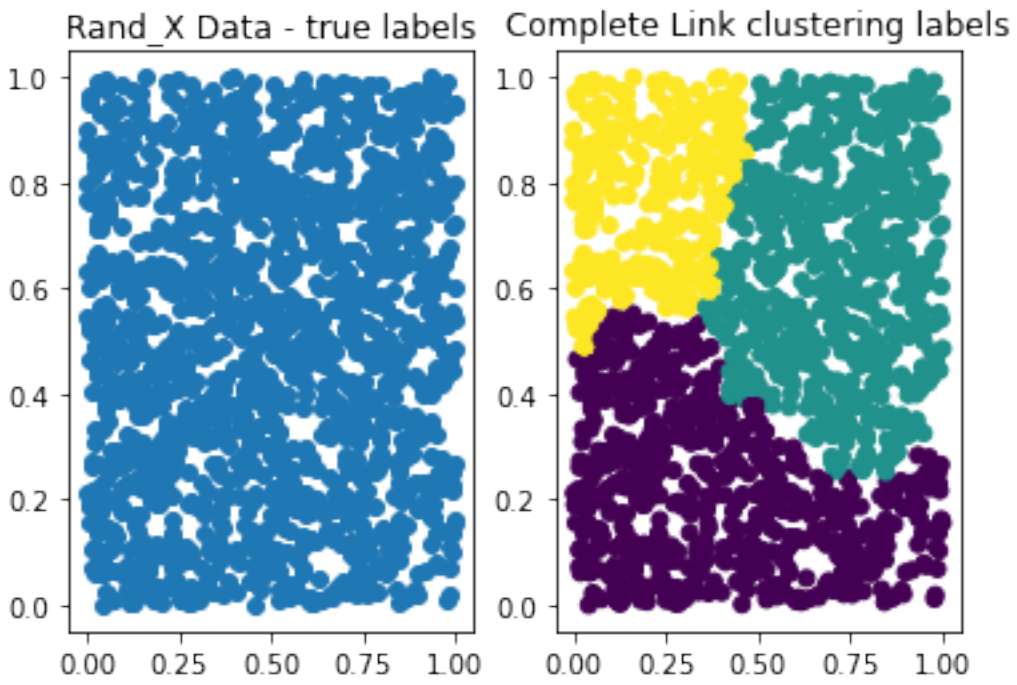
[73]:
```
n_clusters = 3
gmm = GaussianMixture(n_components=n_clusters, covariance_type='full')
y_pred38 = gmm.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred38)
plt.title('Estimation Maximization clustering labels')
plt.show()
```



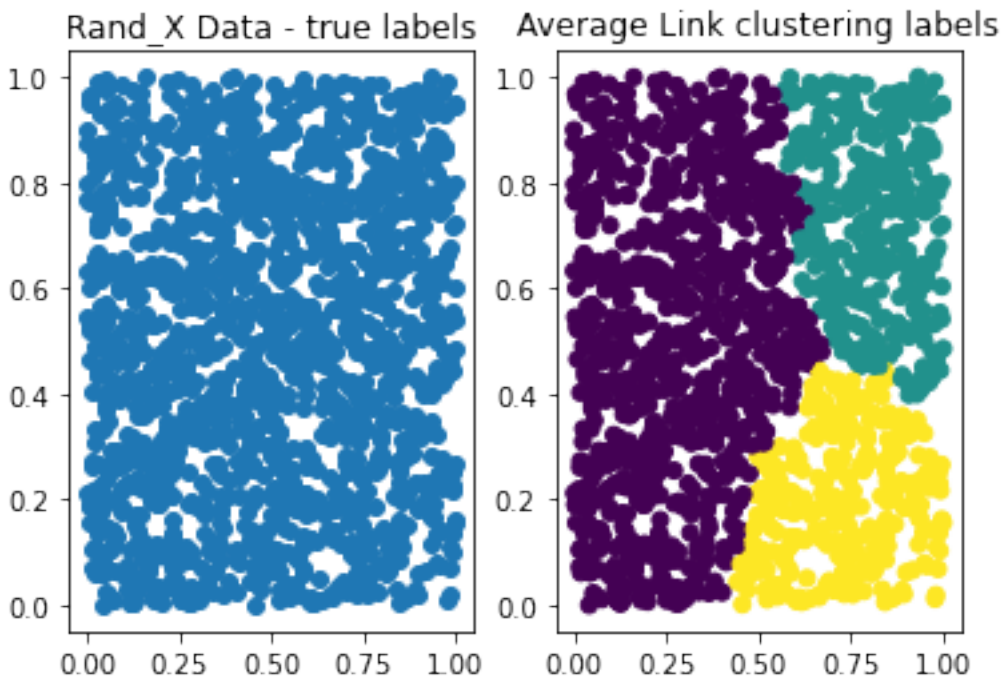
[74]:
```
n_clusters = 3
single_linkage = AgglomerativeClustering(linkage="single",␣
 ↪n_clusters=n_clusters)
y_pred39 = single_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred39)
plt.title('Single Link clustering labels')
plt.show()
```



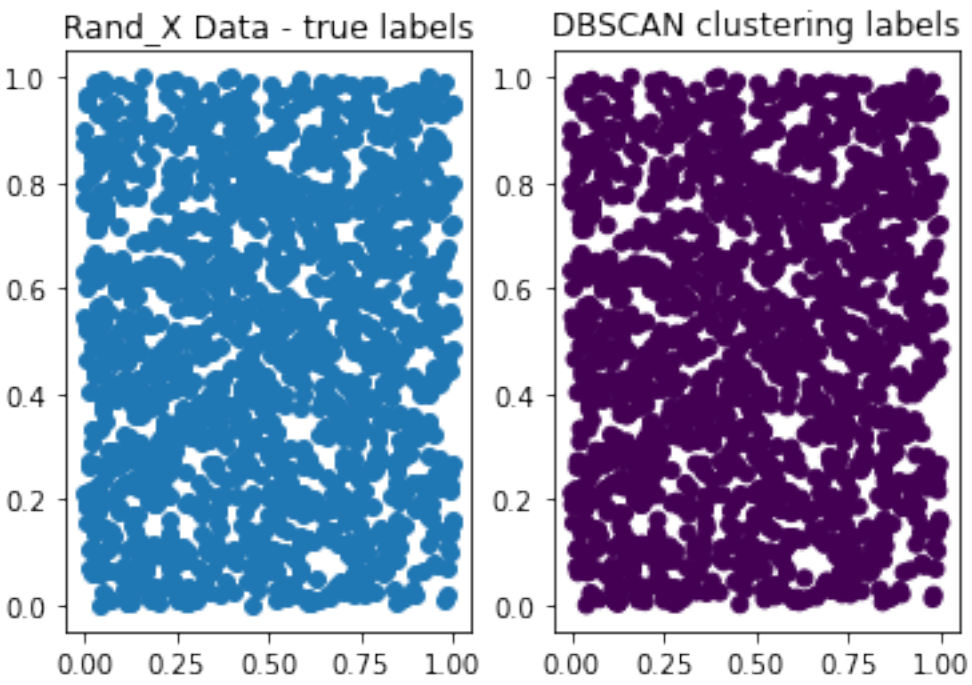
[75]:
```
n_clusters = 3
complete_linkage = AgglomerativeClustering(linkage="complete",␣
 ↪n_clusters=n_clusters)
y_pred40 = complete_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred40)
plt.title('Complete Link clustering labels')
plt.show()
```



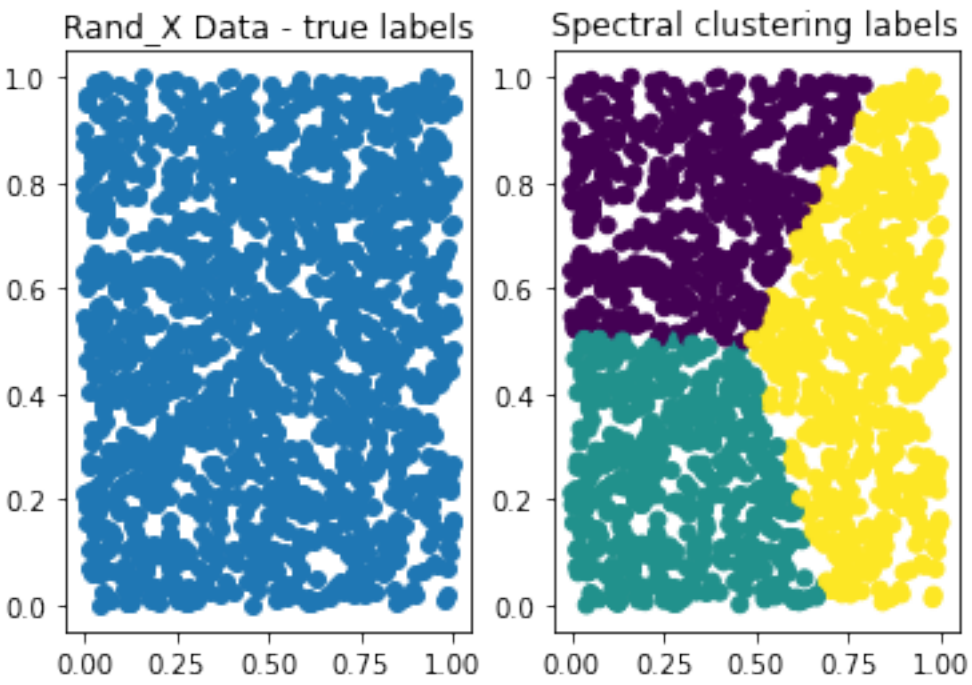
[76]:
```
n_clusters = 3
average_linkage = AgglomerativeClustering(linkage="average",␣
 ↪n_clusters=n_clusters)
y_pred41 = average_linkage.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred41)
plt.title('Average Link clustering labels')
plt.show()
```



[77]:
```
dbscan = DBSCAN(eps=1, min_samples=10)
y_pred41 = dbscan.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred41)
plt.title('DBSCAN clustering labels')
plt.show()
```



[78]:
```
n_clusters = 3
spectral = SpectralClustering(n_clusters=n_clusters, random_state=random_state)
y_pred42 = spectral.fit_predict(Rand_X)
plt.subplot(1,2,1)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1])
plt.title('Rand_X Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Rand_X[:, 0], Rand_X[:, 1], c=y_pred42)
plt.title('Spectral clustering labels')
plt.show()
```



Answer: When we compare DBSCAN with the actual output, it is almost the same because it depens on the value of epsilon and minimum points, so we can adjust the output as per the values. Similarly, for single link the algorithm will look for closest pairs of distance between two clusters, and as the data is random it will give us the results based on density of points or closeness of clusters. It will discover clusters for all other algorithms, but it varies for each method based on it's working.

1.0.9 8. Real-world dataset

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

[79]:
```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The features are:

[80]:
```
cancer.feature_names
```

[80]:
```
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:

[81]:
```
cancer.target_names
```

[81]:
```
array(['malignant', 'benign'], dtype='<U9')
```

Create dataset for classification

[82]:
```
Cancer_X = cancer.data
Cancer_y = cancer.target
```

Size of Cancer_X and Cancer_y

[83]:
```
Cancer_X.shape
```

[83]:
```
(569, 30)
```

[84]:
```
Cancer_y.shape
```

[84]:
```
(569,)
```

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

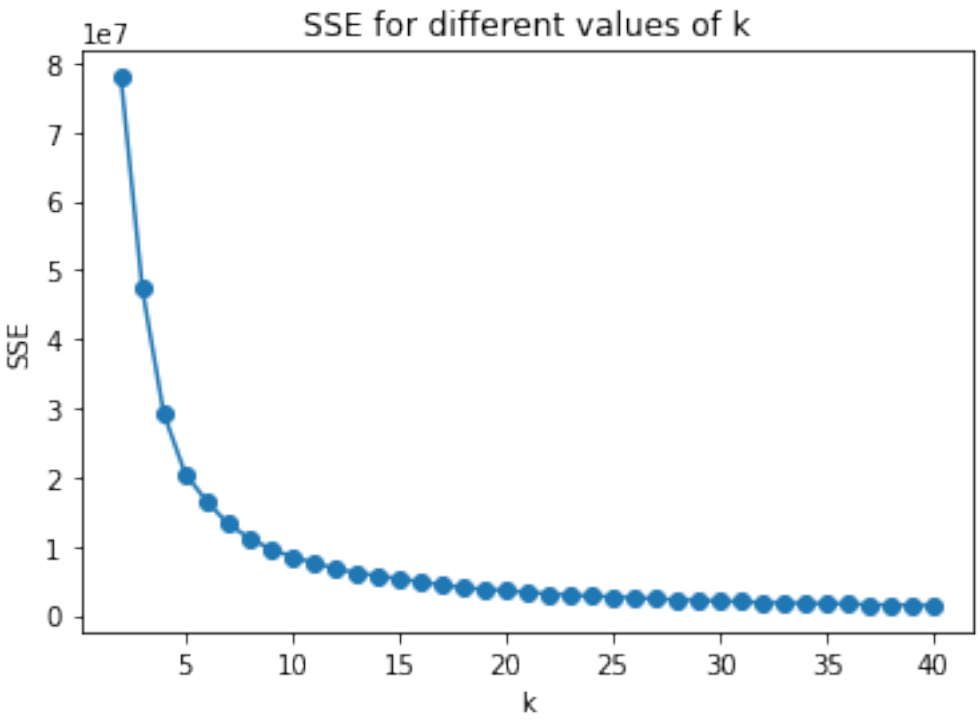
[85]:
```
score = np.zeros(41);
for i in range(2,41):
    kmeans = KMeans(n_clusters=i, random_state=random_state); #Initializing␣
 ↪KMeans for different n_clusters
    kmeans.fit_predict(Cancer_X)  #Clustering using KMeans
    score[i] = -kmeans.score(Cancer_X)  #Computing SSE
    print("SSE for k=",i,":", 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
SSE for k= 40 : 1435813.77
```

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

[86]:
```
plt.plot(range(2,41),score[2:41])
plt.scatter(range(2,41),score[2:41])
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: Based on this plot, we see that beyond k=9 there is no significant reduction in SSE. So we choose k=9.

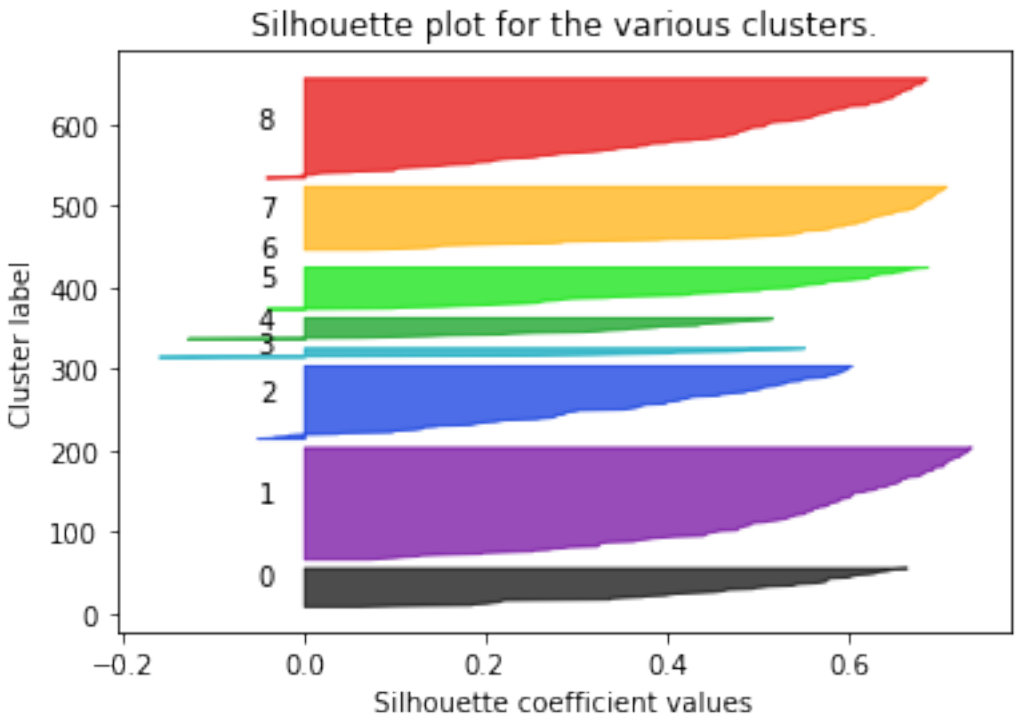
Question 8d: Using the 'k' you chose in (c), compute k-Means clustering.

[87]:
```
n_clusters = 9
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state);
y_pred43 = kmeans.fit_predict(Cancer_X)
score = -kmeans.score(Cancer_X)
print("SSE for k=:", round(score,2))
```

```
SSE for k=: 9609383.58
```

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

[88]:
```
silhouette(Cancer_X, y_pred43)
```



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: Cluster1 seems to be largest of all the clusters and also it is far away from cluster 2, cluster 3 and cluster 4 as its Silhouette value is largerst of all. Clusters which are close together and do not have negative Silhouette coefficient are good clusters such as Cluster1, Cluster7 and Cluster0. All other clusters are having few negative values and are not as good as cluster 1,7 and 0. Clusters 3, 4 and 2 and not good clusters.

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.

[89]:
```
print("Rand Index Value for Cancer_y: ", rand_index(y_pred43, Cancer_y))
```

```
Rand Index Value for Cancer_y:  0.5781888165548652
```

Answer: Cluster are not so good nor bad as rand index value is around 0.6. For good clusters rand index should be around 1.

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:

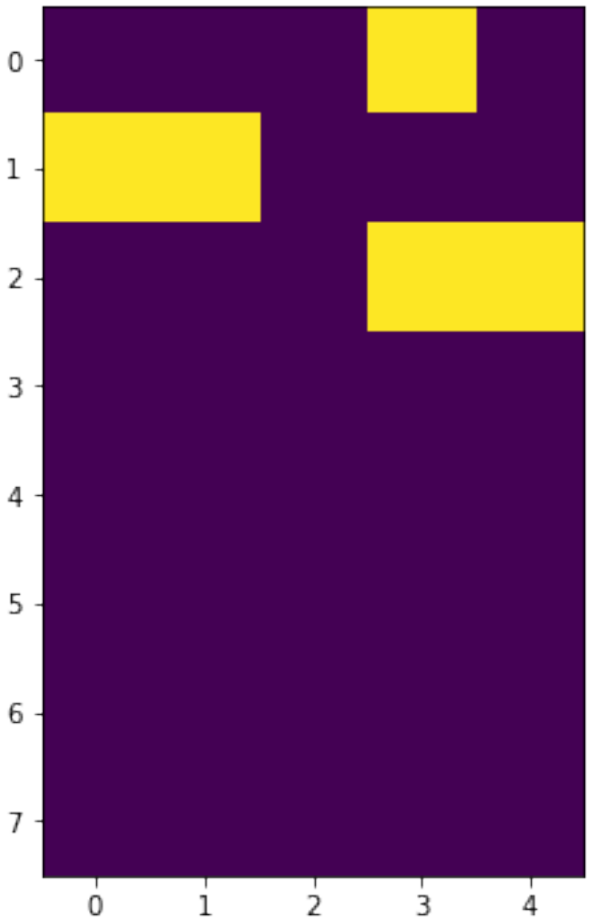
y_pred = dbscan.fit_predict(Cancer_X)

max(y_pred)+1

[90]:
```
eps= [50,100,150,200,250,300,400,500]
minpts = [10,15,20,25,30]
sum2 = []
for i in eps:
    for j in minpts:
        dbscan = DBSCAN(eps=i, min_samples=j)
        y_pred = dbscan.fit_predict(Cancer_X)
        c = max(y_pred)+1
        sum2.append(c)
sum2 = np.reshape(sum2, (8,5))
print("Matrix of number of clusters: \n", sum2)
```

```
Matrix of number of clusters:
 [[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]]
```

[91]:
```
plt.figure(figsize=(6,6))
plt.imshow(sum2)
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)

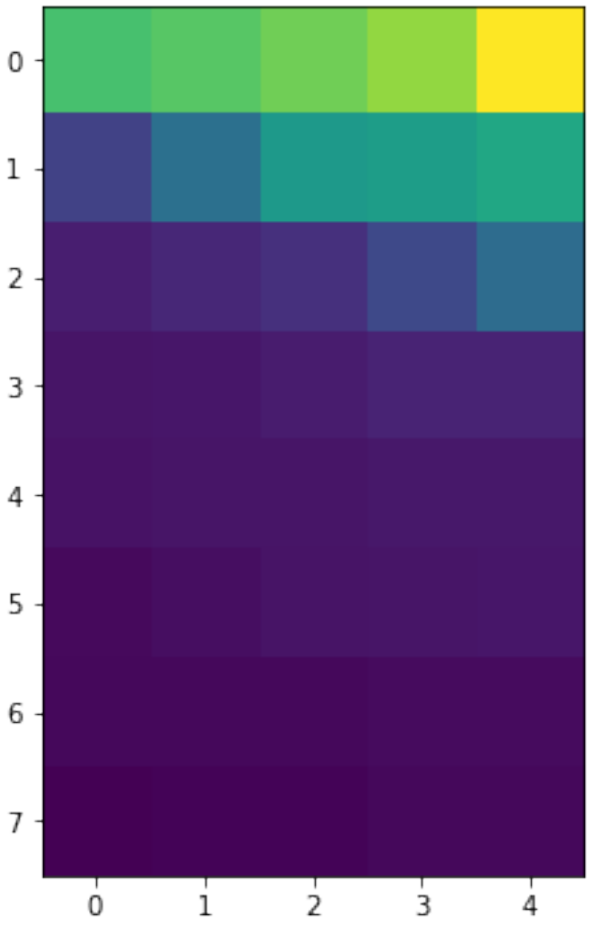
sum(y_pred==-1)

[92]:
```
eps= [50,100,150,200,250,300,400,500]
minpts = [10,15,20,25,30]
sum1 = []
for i in eps:
    for j in minpts:
        dbscan = DBSCAN(eps=i, min_samples=j)
        y_pred = dbscan.fit_predict(Cancer_X)
        c =  np.sum(y_pred==-1)
        sum1.append(c)
```

```
sum1 = np.reshape(sum1, (8,5))
print("Matrix of noise points: \n", sum1)
```

```
Matrix of noise points:
 [[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]]
```

[93]:
```
plt.figure(figsize=(6,6))
plt.imshow(sum1)
plt.show()
```



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: Based on the number of clusters matrix atmost 2 clusters should be present in R^d. Also, if the minpoints are higher in number then the noise points are increasing for this data.

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.

[94]:
```
dbscan = DBSCAN(eps=50, min_samples=30)
y_pred44 = dbscan.fit_predict(Cancer_X)
c =  np.sum(y_pred44==-1)
c
```

[94]:
```
262
```

[95]:
```
print("Rand Index Value for Cancer_y: ", rand_index(abs(y_pred44), Cancer_y))
```

```
Rand Index Value for Cancer_y:  0.757914799871284
```

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: When using K-Means we're getting a low Rand Index compared to DBSCAN algorithm, because DBSCAN is based on densities of points. Also, we should use DBSCAN for R^d space as K-Means will only work for datasets with low dimentionality.