M11-L1 Problem 3

In this problem you will use the `sklearn` implementation of hierarchical clustering with three different linkage criteria `('single', 'complete', 'average')` to clusters two datasets: a "blob" shaped dataset with three classes, and a concentric circle dataset with two classes.

In [25]:

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

from sklearn.datasets import make_blobs, make_circles
from sklearn.cluster import AgglomerativeClustering

## DO NOT MODIFY
def plotter(x, labels = None, ax = None, title = None):
    if ax is None:
        _, ax = plt.subplots(dpi = 150, figsize = (4,4))
        flag = True
    else:
        flag = False
    for i in range(len(np.unique(labels))):
        ax.scatter(x[labels == i, 0], x[labels == i, 1], alpha = 0.5)
    ax.set_xlabel('$x_0$')
    ax.set_ylabel('$x_1$')
    ax.set_aspect('equal')
    if title is not None:
        ax.set_title(title)
    if flag:
        plt.show()
    else:
        return ax
```

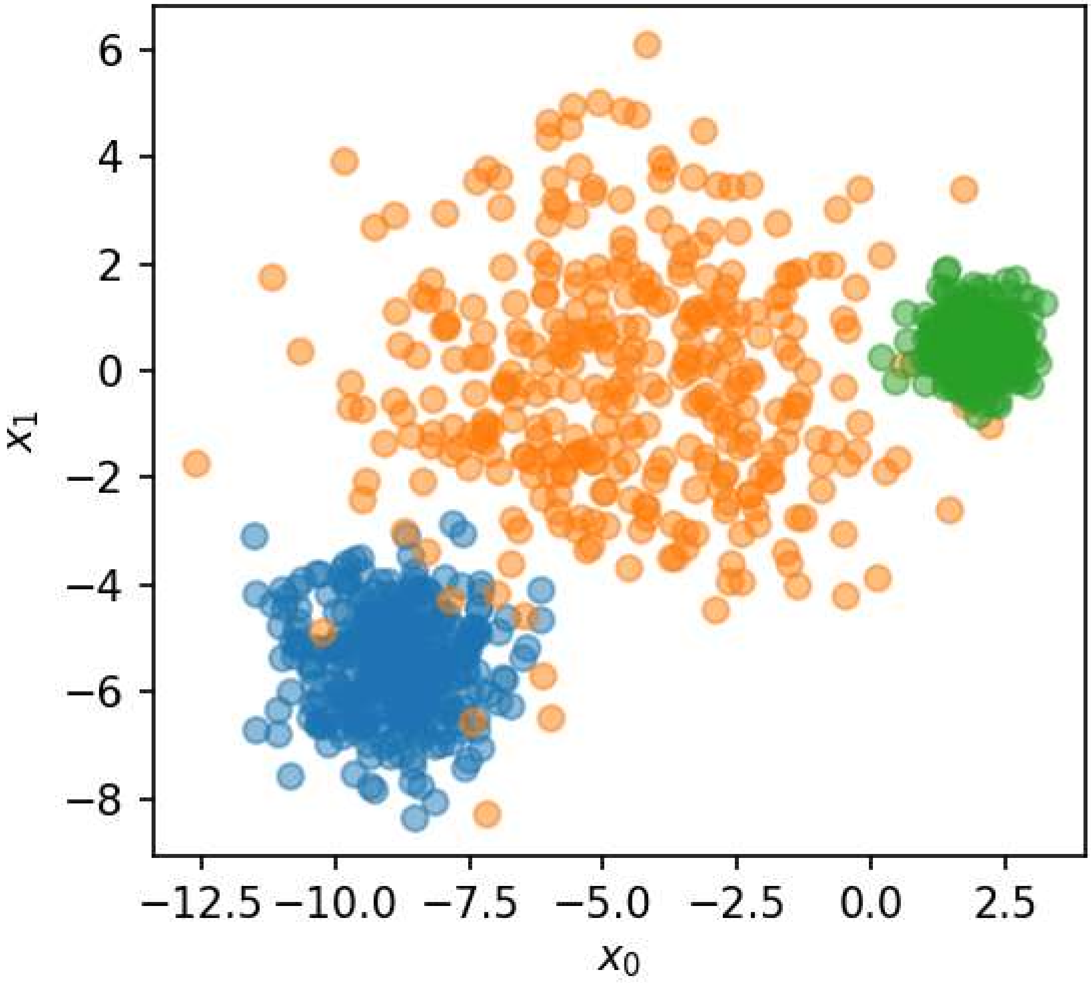
First we will consider the "blob" dataset, generated below. Visualize the data using the provided `plotter(x, labels)` function.

In [26]:

```
## DO NOT MODIFY
x, labels = make_blobs(n_samples = 1000, cluster_std=[1.0, 2.5, 0.5], random_state
```

In [27]:

```
## YOUR CODE GOES HERE
plotter(x,labels)
```

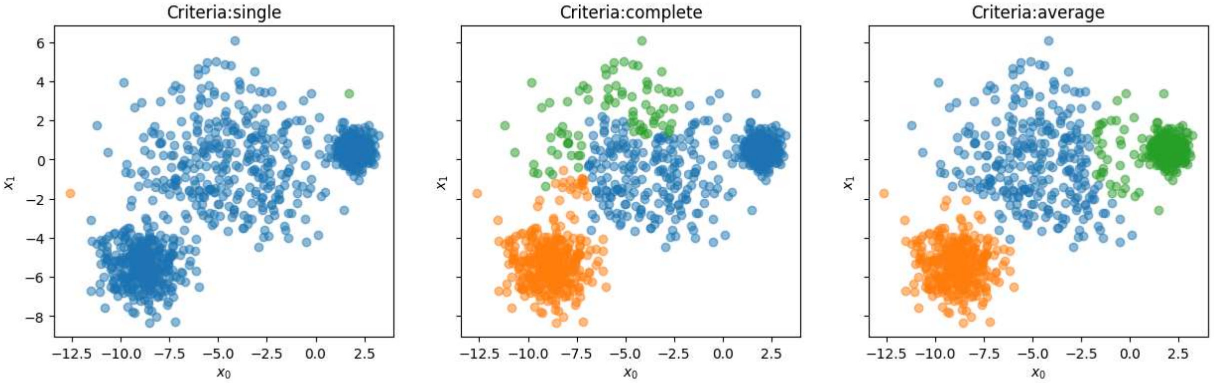


Using the `AgglomerativeClustering()` function, generate 3 side-by-side plots using `plt.subplots()` and the provided `plotter(x, labels, ax, title)` function to visualize the results of the following three linkage criteria `['single', 'complete', 'average']`.

Note: the `plt.subplots()` function will return `fig, ax`, where `ax` is an array of all the subplot axes in the figure. Each individual subplot can be accessed with `ax[i]` which you can then pass to the `plotter()` function's `ax` argument.

In [28]:

```
## YOUR CODE GOES HERE
criterias = ['single', 'complete', 'average']
fig, ax = plt.subplots(1, len(criterias), figsize=(15, 5), sharey=True)
for i,c in enumerate(criterias):
    clustering = AgglomerativeClustering(linkage = c,n_clusters = 3).fit(x)
    labels= clustering.labels_
    title = f'Criteria:{c}'
    plotter(x,labels,ax[i],title)
```



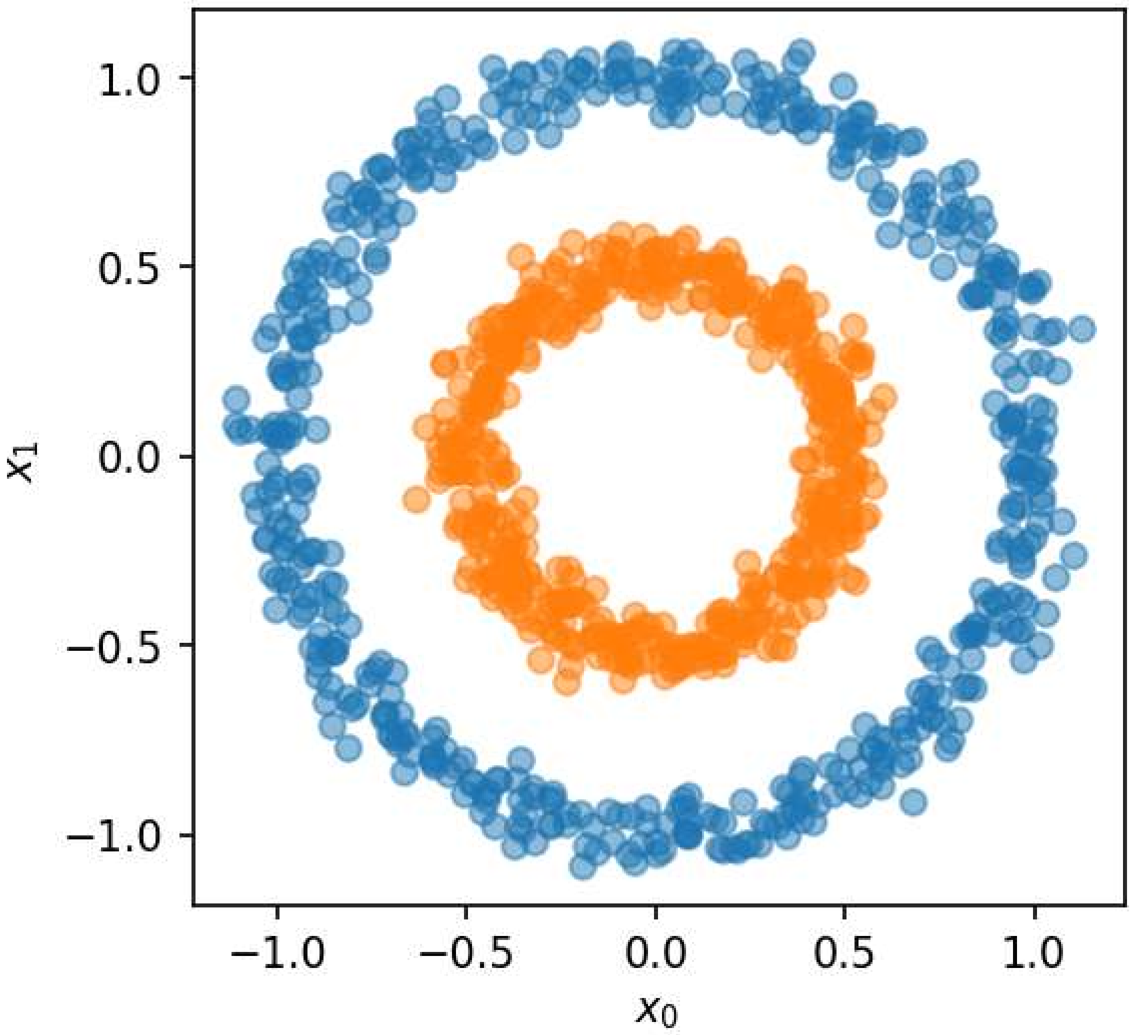
Now we will work on the concentric circle dataset, generated below. Visualize the data using the provided `plotter(x, labels)` function.

In [29]:

```
## DO NOT MODIFY
x, labels = make_circles(1000, factor = 0.5, noise = 0.05, random_state = 0)
```

In [30]:

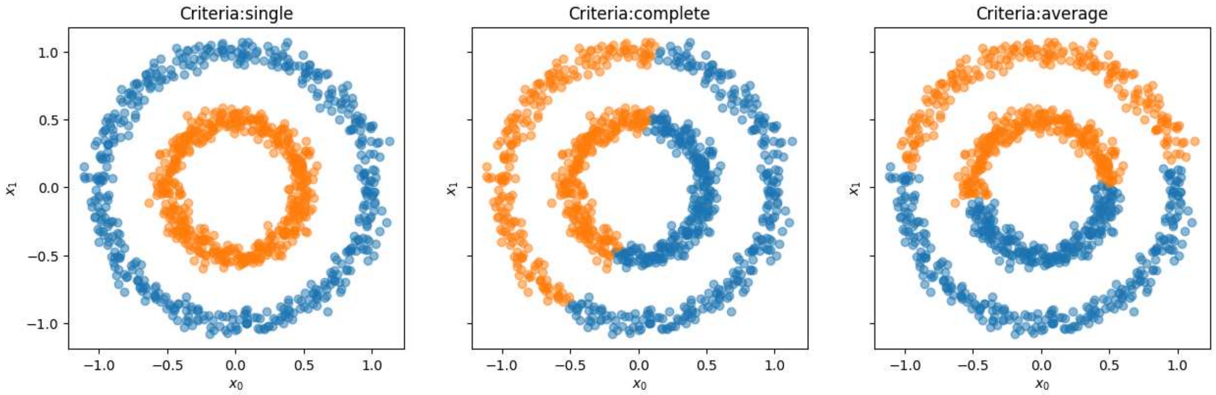
```
## YOUR CODE GOES HERE
plotter(x,labels)
```



Again, use the `AgglomerativeClustering()` function to generate 3 side-by-side plots using `plt.subplots()` and the provided `plotter(x, labels, ax, title)` function to visualize the results of the following three linkage criteria `['single', 'complete', 'average']` for the concentric circle dataset.

In [31]:

```
## YOUR CODE GOES HERE
criterias = ['single', 'complete', 'average']
fig, ax = plt.subplots(1, len(criterias), figsize=(15, 5), sharey=True)
for i,c in enumerate(criterias):
    clustering = AgglomerativeClustering(linkage = c,n_clusters = 2).fit(x)
    labels= clustering.labels_
    title = f'Criteria:{c}'
    plotter(x,labels,ax[i],title)
```



Discussion

Discuss the performance of the three different linkage criteria on the "blob" dataset, and then on the concentric circle dataset. Why do some linkage criteria perform better on one dataset, but worse on others?

Your response goes here

Average creteria performs the best on the blob dataset. It considers the average distance between single and complete linkages, so the cluster it created is compact and does not have chaining effect.

The single performs the best on the concentric circles due to its chainging effect.