Imbalanced data:

One group will have a lot more data points than the other two combined.

In [5]:
```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#For Jupyter Notebooks to show the plots
%matplotlib inline
```

In [46]:
```
# Create data from three different multivariate distributions
X_1 = np.random.multivariate_normal(mean=[4, 0], cov=[[1, 0], [0, 1]], size=75
)
X_2 = np.random.multivariate_normal(mean=[6, 6], cov=[[2, 0], [0, 2]], size=50
0)
X_3 = np.random.multivariate_normal(mean=[1, 5], cov=[[1, 0], [0, 2]], size=20
)
```

In [47]:
```
X_1.shape
```

Out[47]:
```
(75, 2)
```

In [48]:
```
X = np.concatenate([X_1, X_2, X_3])
```

In [49]:
```
X.shape
```

Out[49]:
```
(595, 2)
```

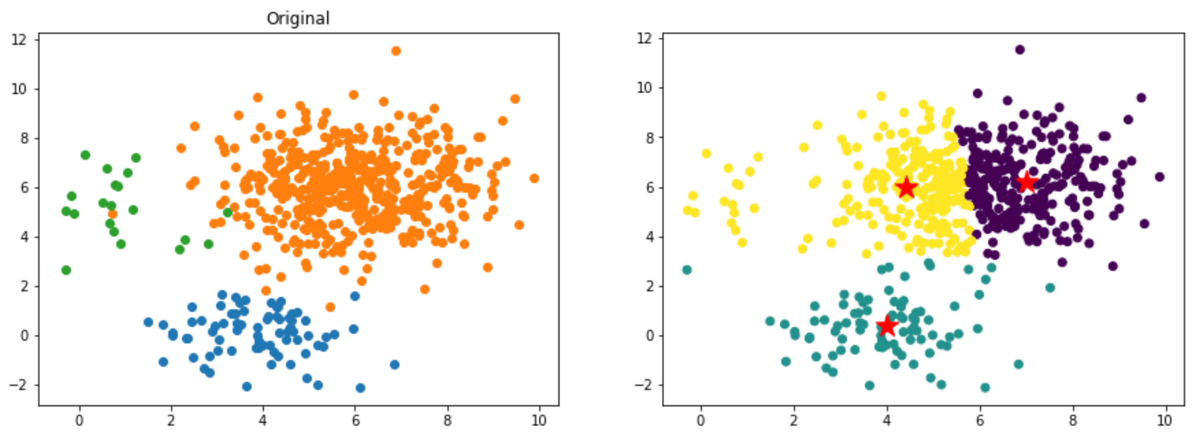
In [57]:
```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(X)
```

Out[57]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [58]:
```
labels = model.labels_
centroids = model.cluster_centers_
```

```
In [59]: fig, ax = plt.subplots(1,2, figsize=(15,5))
         ax[0].scatter(X_1[:,0], X_1[:,1])
         ax[0].scatter(X_2[:,0], X_2[:,1])
         ax[0].scatter(X_3[:,0], X_3[:,1])
         ax[0].set_title('Original')

         ax[1].scatter(X[:,0], X[:,1], c = labels)
         ax[1].scatter(centroids[:,0], centroids[:,1], marker ='*', s =300, c='r')
```

```
Out[59]: <matplotlib.collections.PathCollection at 0x1f5c9ef0208>
```



Looks like kmeans couldn’t figure out the clusters correctly. Since it tries to minimize the within-cluster variation, it gives more weight to bigger clusters than smaller ones. In other words, data points in smaller clusters may be left away from the centroid in order to focus more on the larger cluster.

```
In [ ]:
```

Data sets with complicated geometric shapes

```
In [92]: from sklearn.datasets import make_circles, make_moons
```

Cricles

Return:

X : array of shape [n_samples, 2] The generated samples.

y : array of shape [n_samples] The integer labels (0 or 1) for class membership of each sample.

```
In [96]: # Cricles
         X1 = make_circles(factor=0.5, noise=0.05, n_samples=1500)
```

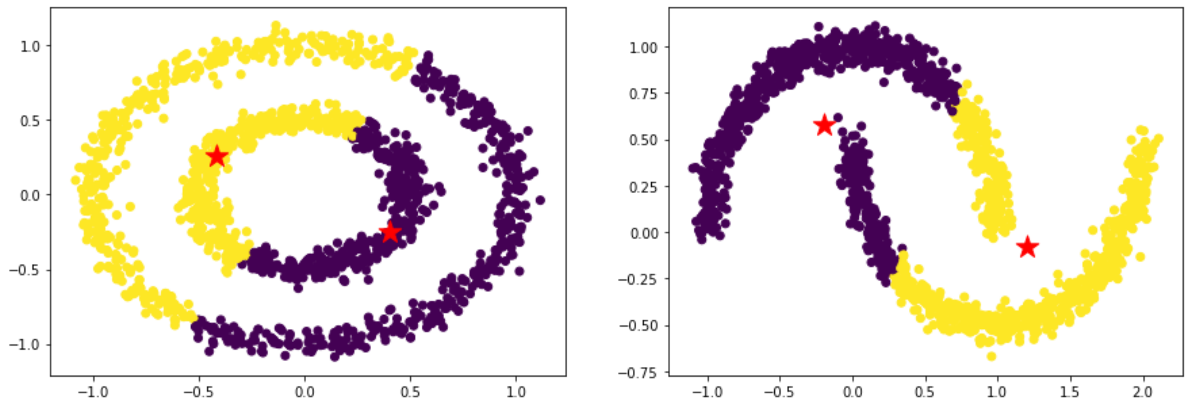
```
In [97]: len(X1)
```

```
Out[97]: 2
```

In [98]:
```
# Moons
X2 = make_moons(n_samples=1500, noise=0.05)
len(X2)
```

Out[98]:
```
2
```

In [106]:
```
fig, ax = plt.subplots(1,2, figsize = (15,5))

for i, X in enumerate([X1, X2]):

    model = KMeans(n_clusters=2)
    model.fit(X[0])
    labels = model.labels_
    centroids = model.cluster_centers_

    ax[i].scatter(X[0][:,0], X[0][:,1], c=labels)
    ax[i].scatter(centroids[:,0], centroids[:,1], marker ='*', s =300, c='r')
```

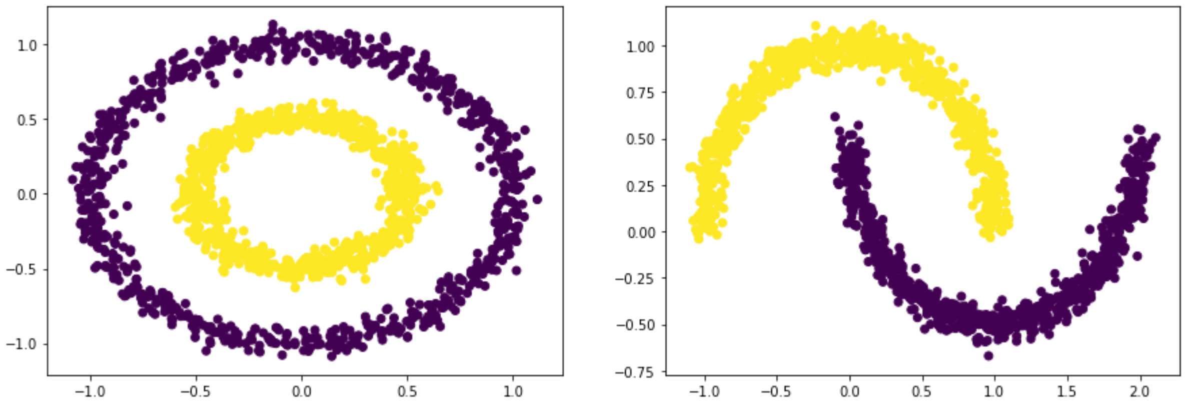


Solution: SpectralClustering

In [107]:
```
from sklearn.cluster import SpectralClustering
```

```
In [111]: fig, ax = plt.subplots(1,2, figsize = (15,5))

          for i, X in enumerate([X1, X2]):

              model = SpectralClustering(n_clusters=2, affinity='nearest_neighbors')
              model.fit(X[0])

              labels = model.labels_

              ax[i].scatter(X[0][:,0], X[0][:,1], c=labels)
```

```
C:\Users\FirouzehPC\Anaconda3\lib\site-packages\sklearn\manifold\spectral_emb
edding_.py:237: UserWarning: Graph is not fully connected, spectral embedding
may not work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
C:\Users\FirouzehPC\Anaconda3\lib\site-packages\sklearn\manifold\spectral_emb
edding_.py:237: UserWarning: Graph is not fully connected, spectral embedding
may not work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```



```
In [ ]:
```