K.SAI SIDHARTHA

PG -DBDA

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
```

- Importing the libraries which are needed
- pandas - for reading data
- numpy - for linear algebra
- matplotlib for plots (seaborn too)
- make blobs for creating clustter objects for kmeans algorithm

In [2]:

```
x,y = make_blobs(n_samples = 1000, centers=4, n_features=2)
```

- making blobs named (x,y) which are nothing but clusters

In [3]:

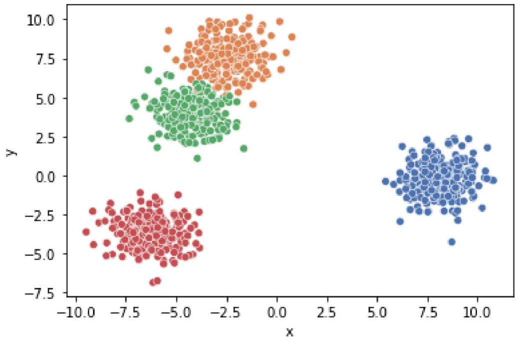
```
x
```

Out[3]:

```
array([[-6.67561614, -3.4435433 ],
       [-3.02953208,  3.30961756],
       [ 6.5885794 , -1.1333549 ],
       ...,
       [-3.46081773,  3.35737387],
       [ 7.38636377,  0.24428236],
       [ 8.9771507 , -0.12480738]])
```

In [5]:

```
sns.scatterplot(x = [X[0] for X in x], y = [X[1] for X in x], hue = y, palette='deep',legend= None)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



The above Scatter plots indicates how are the random points placed and we can see 4 clusters in it

In [6]:

```
m = x.shape[0]
n = x.shape[1]
```

- Here , for future comfort we have assigned the shapes of arrays for 2 variables called M an N

In [7]:

```
import random
```

- In kmeans we hereby dissolve random variables we have imported random library which is a part of numpy

STEPS FOR K MEANS CLUSSTERING

1. scale the data
2. initialize the centroids
3. Label data Points
4. update each centroid

5. repeat steps 3 and 4 until centroid stops changing

```
In [8]: centroids = np.array([]).reshape(n,0)
```

- STEP NO 2 : INITAILZING THE CENTROIDS (as we knew we can calculate the distance between the points from centroids in order to there we declare centroids)

```
In [9]: centroids

Out[9]: array([], shape=(2, 0), dtype=float64)
```

```
In [10]: k=5
         for i in range(k):
             centroids = np.c_[centroids,x[random.randint(0,m-1)]]
```

```
In [11]: centroids

Out[11]: array([[-1.7064217 , -2.09812574, -4.08801443, -3.26238424, -2.06911663],
                [ 8.16967838,  7.32936132,  3.43360399,  4.01502487,  9.8292304 ]])
```

```
In [12]: euclid = np.array([]).reshape(m,0)
```

- The distance between the points can be calculated with the help of euclid distace formula (maths formula), the STEP3

```
In [13]: for i in range(k):
             dist = np.sum((x-centroids[:,i])**2,axis=1)
             euclid=np.c_[euclid,dist]
```

```
In [14]: minimum = np.argmin(euclid,axis=1)+1
```

```
In [15]: cent = {}
         for i in range(k):
             cent[i+1] = np.array([]).reshape(2,0)
```

- Step 4 : updating the centroids

```
In [16]: for i in range(m):
             cent[minimum[i]]=np.c_[cent[minimum[i]],x[i]]
         for i in range(k):
             cent[i+1]=cent[i+1].T
```

```
In [17]: cent

        [-1.94559692,  6.78152568],
        [-3.28298685,  6.64284013],
        [-2.09343968,  7.78321022],
        [-3.19335001,  6.14437182],
        [-2.88081618,  7.66809217],
        [-3.91599365,  7.35685411],
        [-3.58254062,  7.75037147],
        [-3.0117779 ,  6.86963237],
        [-2.15935369,  7.52379568],
        [-2.47541568,  6.58053593],
        [-1.42523325,  6.52258453],
        [-3.07720034,  6.66548566],
        [-1.40345516,  5.73224724],
        [-3.33736264,  7.95943543],
        [ 9.78557157,  1.44741459],
        [-1.17736613,  6.19572496],
        [-3.72793639,  8.55156914],
        [-3.24890951,  6.38428715],
        [-3.35658228,  7.91746408],
        [-2.05593876,  7.21705921],
```

```
In [19]: for i in range(k):
             centroids[:,i] = np.mean(cent[i+1],axis=0)
```

```
In [21]: centroids[:,i]

Out[21]: array([-2.42545325,  9.50210648])
```

- The updated centroids

- THE FINAL STEP 5: WHERE WE COMBINE EACH AND EVERY THING AT LAST TO GET FINAL CENTROIDS OF THE DATA POINTS

In [24]:

```
n_iters = 50
for i in range(n_iters):
    euclid=np.array([]).reshape(m,0)

    for p in range(k):
        dist=np.sum((x-centroids[:,p])**2,axis=1)
        euclid = np.c_[euclid,dist]
    c = np.argmin(euclid,axis=1)+1
    cent={}
    for p in range(k):
        cent[p+1]=np.array([]).reshape(2,0)
    for p in range(m):
        cent[c[p]]=np.c_[cent[c[p]],x[p]]

    for p in range(k):
        cent[p+1]=cent[p+1].T
    for p in range(k):
        centroids[:,p]=np.mean(cent[p+1],axis=0)
    final=cent
```

In [25]:

```
final
```

Out[25]:

```
{1: array([], shape=(0, 2), dtype=float64),
 2: array([[-6.67561614, -3.4435433 ],
        [-3.02953208,  3.30961756],
        [ 6.5885794 , -1.1333549 ],
        ...,
        [-3.46081773,  3.35737387],
        [ 7.38636377,  0.24428236],
        [ 8.9771507 , -0.12480738]]),
 3: array([], shape=(0, 2), dtype=float64),
 4: array([], shape=(0, 2), dtype=float64),
 5: array([], shape=(0, 2), dtype=float64)}
```

In [ ]:

In [ ]: