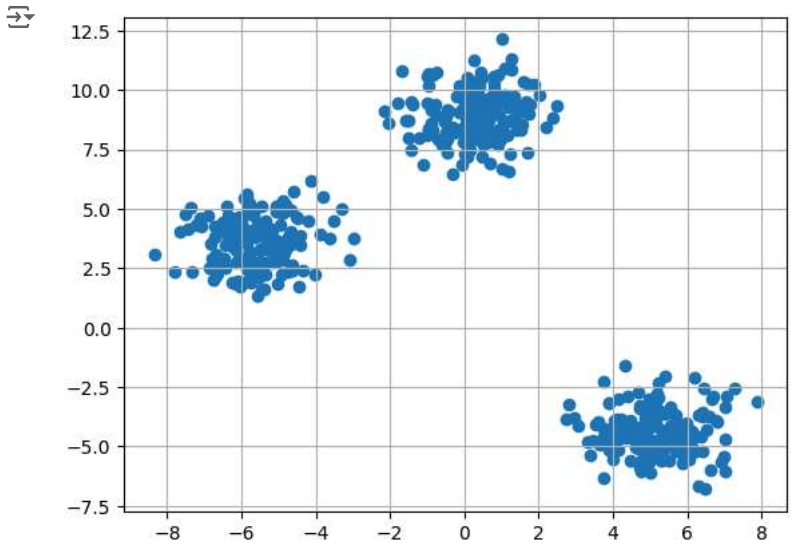
Name=Md Arham Tabib Id=221-15-5707 section= 61_K1 ==comment

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.datasets as make_blobs
import sklearn.datasets as datasets
```

creating custom dataset and ploting data set

you can change n_samples, centers

```
x, y = datasets.make_blobs(n_samples=500, n_features=2, centers=3, random_state=23)
fig=plt.figure(0)
plt.grid(True)
plt.scatter(x[:, 0], x[:, 1])
plt.show()
```



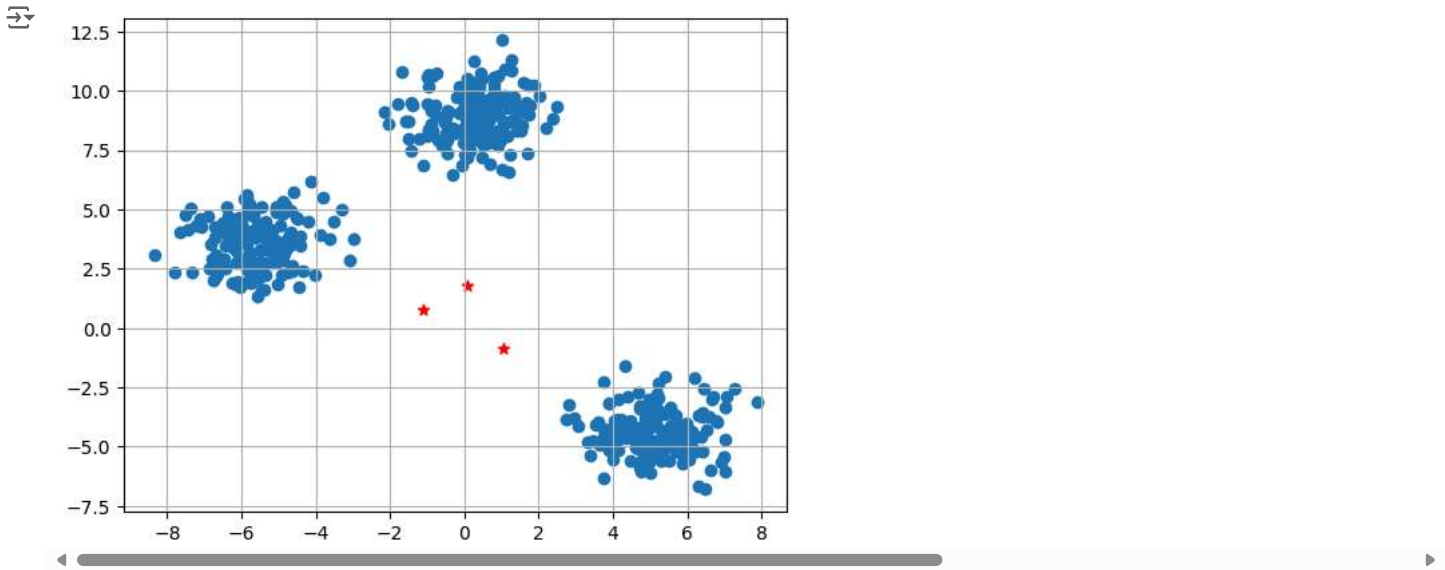
initialize the random centroids

the k value will be equal to ur centers

```
k=3
clusters={}
np.random.seed(23)
for idx in range(k):
  center=2*(2*np.random.random((x.shape[1],))-1)
  points=[]
  cluster={
      'center':center,
      'points':[]
  }
  clusters[idx]=cluster
clusters
```

```
{0: {'center': array([0.06919154, 1.78785042]), 'points': []},
 1: {'center': array([ 1.06183904, -0.87041662]), 'points': []},
 2: {'center': array([-1.11581855,  0.74488834]), 'points': []}}
```

```
plt.scatter(x[:,0],x[:,1])
plt.grid(True)
for i in clusters:
  center=clusters[i]['center']
  plt.scatter(center[0],center[1],marker='*',color='red')
plt.show()
```



define euclidean distance

```
def distance(p1,p2):
  return np.sqrt(np.sum((p1-p2)**2))
```

create function to assign and update the distance

```
def assign(x,clusters):
  for idx in range(x.shape[0]):
    dist=[]
    curr_x=x[idx]
    for i in range(k):
     dis=distance(curr_x,clusters[i]['center'])
     dist.append(dis)
    curr_cluster=np.argmin(dist)
    clusters[curr_cluster]['points'].append(curr_x)
  return clusters


def update_clusters(x,clusters):
  for i in range(k):
    points=np.array(clusters[i]['points'])
    if points.shape[0] > 0:
      new_center=np.mean(points,axis=0)
      clusters[i]['center']=new_center
      clusters[i]['points']=[]
  return clusters
```

create the function to predict the cluster for datapoints

```
def pred_cluster(x,clusters):
  pred=[]
  for i in range(x.shape[0]):
    dist=[]
    for j in range(k):
      dist.append(distance(x[i],clusters[j]['center']))
    pred.append(np.argmin(dist))
  return pred
```

Assign,update, and predict tghe cluster center

```
clusters=assign(x,clusters)
clusters=update_clusters(x,clusters)
pred=pred_cluster(x,clusters)
```

plot data with their predicted cluster center

```
plt.scatter(x[:,0],x[:,1],c=pred)
for i in clusters:
  center=clusters[i]['center']
  plt.scatter(center[0],center[1],marker='^',color='red')
plt.show()
```