Worksheet 07

Name: Rishab Sudhir

UID: U648191615

Topics

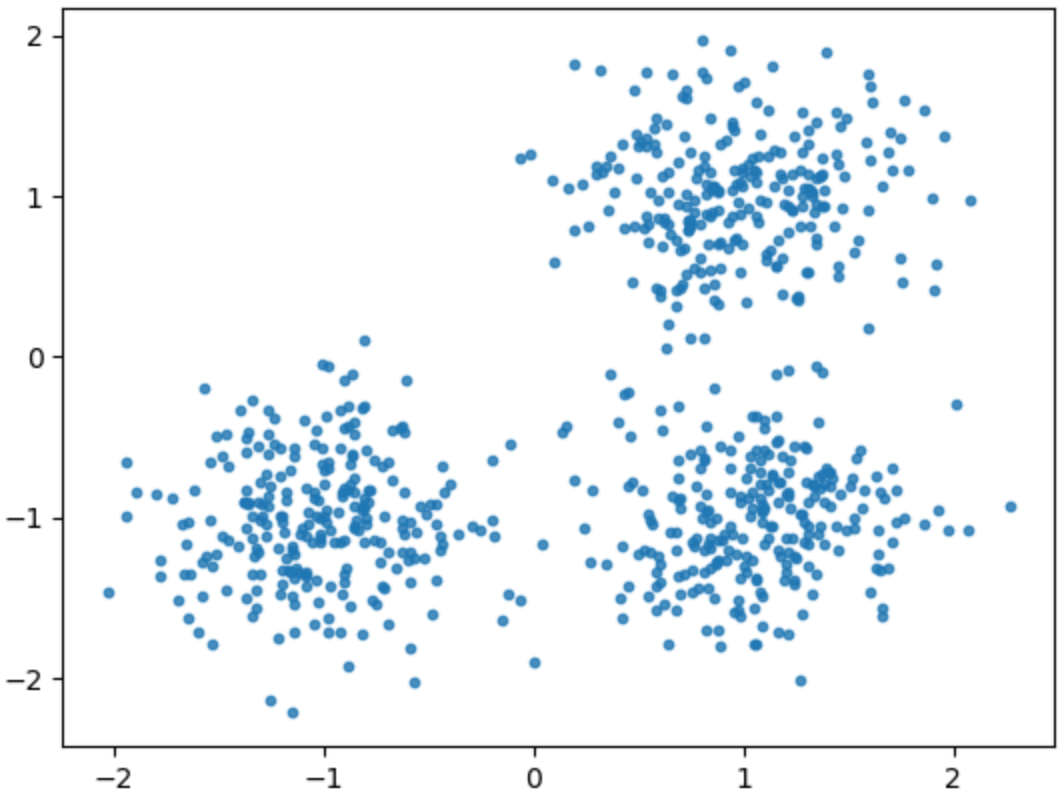
- Density-Based Clustering

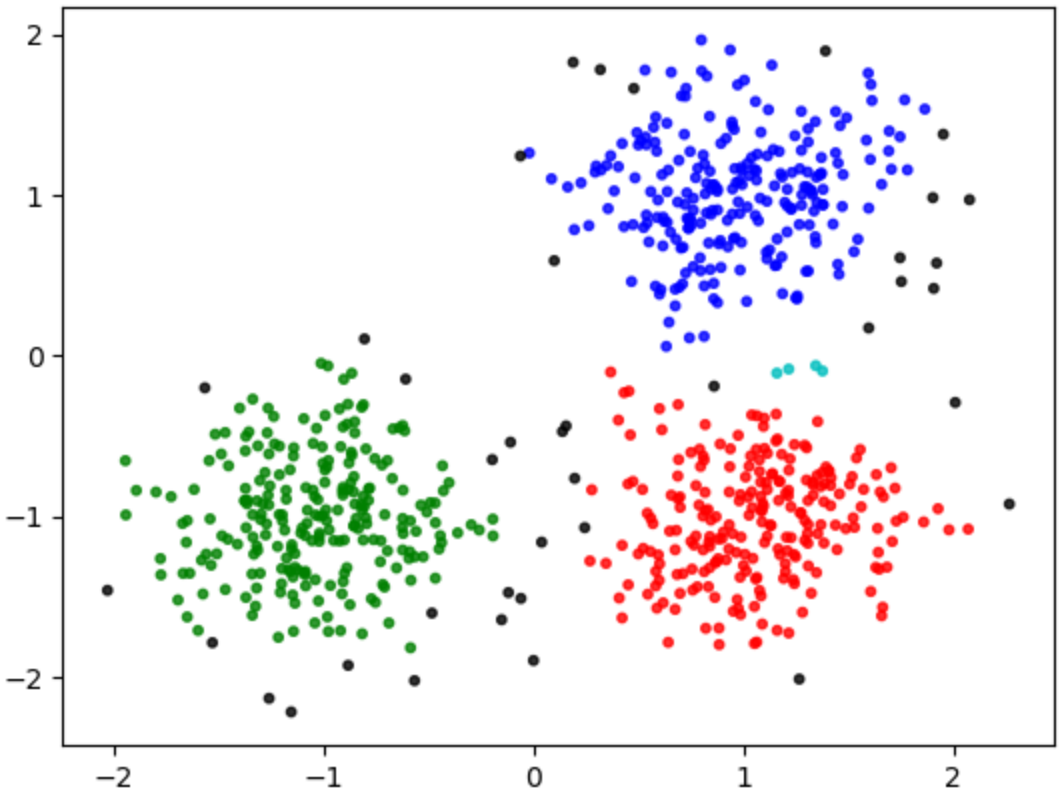
Density-Based Clustering

Follow along with the live coding of the DBScan algorithm.

In [27]:
```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.datasets as datasets

centers = [[1, 1], [-1, -1], [1, -1]]
X, _ = datasets.make_blobs(n_samples=750, centers=centers, cluster_std=0
                           random_state=0)
plt.scatter(X[:,0],X[:,1],s=10, alpha=0.8)
plt.show()

class DBC():

    def __init__(self, dataset, min_pts, epsilon):
        self.dataset = dataset
        self.min_pts = min_pts
        self.epsilon = epsilon
        self.assignments = [-1 for _ in range(len(self.dataset))]

    def distance(self,i,j):
        return np.linalg.norm(self.dataset[i] - self.dataset[j])


    def is_unassigned(self, i):
        return self.assignments[i] == -1

    def is_core(self, i):
        return len(self.get_neighborhood(i)) >= self.min_pts

    def get_neighborhood(self,i):
        neighborhoods = []
        for j in range(len(self.dataset)):
            if i != j and self.distance(i,j) <= self.epsilon:
                neighborhoods.append(j)
        return neighborhoods


    def get_unassigned_neighborhood(self,i):
        neighborhood = self.get_neighborhood(i)
        return [point for point in neighborhood if self.is_unassigned(po

    def make_cluster(self, i, clusterNum):
        self.assignments[i] = clusterNum
        neighborhood_queue = self.get_neighborhood(i)

        while neighborhood_queue:
            next_pt = neighborhood_queue.pop()

            if not self.is_unassigned(next_pt):
                continue

            self.assignments[next_pt] = clusterNum

            if self.is_core(next_pt):
                neighborhood_queue += self.get_unassigned_neighborhood(n

        return
```

```
    def dbscan(self):
        """
        returns a list of assignments. The index of the
        assignment should match the index of the data point
        in the dataset.
        """

        clusterNum = 0

        for i in range(len(self.dataset)):
            if self.assignments[i] != -1:
                continue

            if self.is_core(i):
                # start building a new cluster
                self.make_cluster(i, clusterNum)
                clusterNum += 1

        return self.assignments

clustering = DBC(X, 3, .2).dbscan()
colors = np.array([x for x in 'bgrcmykbgrcmykbgrcmykbgrcmyk'])
colors = np.hstack([colors] * 100)
plt.scatter(X[:, 0], X[:, 1], color=colors[clustering].tolist(), s=10, a
plt.show()
```





In [ ]: