DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a popular clustering algorithm used in machine learning and data mining.
- It is well-suited for identifying clusters of varying shapes and sizes in large spatial databases and can handle noise (outliers) effectively.
- DBSCAN forms clusters based on the density of data points in a given region. A region with a high density of points is considered a cluster, while areas with low density are treated as noise or outliers.
- The algorithm uses 2 parameters
  First parameter ε (epsilon) to define the radius of the neighborhood around each point. This radius determines which points are considered neighbors.
  A second parameter, `min_samples`, specifies the minimum number of points required to form a dense region (core point). If a point has at least `min_samples` points (including itself) within its ε neighborhood, it is considered a core point.

- **Core Points:** Points that have at least `min_samples` points within their ε neighborhood.
- **Border Points:** Points that are within the ε neighborhood of a core point but do not have enough points to be a core point themselves.
- **Noise Points:** Points that are not within the ε neighborhood of any core points and are considered outliers.

Working of DBSCAN

1. **Identify Core Points:** For each point in the dataset, the algorithm counts the number of points within its ε neighborhood. If the count is greater than or equal to `min_samples`, the point is labeled as a core point.
2. **Form the Clusters:**
   - Core points that are within ε distance of each other are grouped together to form a cluster.
   - Border points are added to the nearest core point's cluster.
   - Noise points are identified and ignored for clustering purposes.
3. **Expand Clusters:**
   - Starting with a core point, the algorithm recursively visits all points in its ε neighborhood, adding them to the cluster if they are also core or border points. This process continues until all reachable points have been visited and assigned to a cluster.

Advantages of DBSCAN

- Unlike other clustering algorithms like K-means, DBSCAN can find clusters of arbitrary shapes and sizes.
- Does not require specifying the number of clusters in advance, as opposed to K-means which requires the number of clusters (`k`) to be defined beforehand.
- Can handle noise and outliers effectively by labeling them as noise points, ensuring they do not affect the clustering results.

PROBLEM: Apply the DBSCAN algorithm to the given data points and create the clusters with min_samples = 4 and epsilon (ϵ) = 1.9 p1 (3,7) p2(4,6) p3(5,5) p4(6,4) p5(7,3) p6(6,2) p7(7, 2) p8(8,4) p9(3,3) p10(2,6) p11(3,5) p12(2,4)

Solution: Calculate the distance between each points using Euclidian distance

$$Distance(A(x_1, y_1), B(x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

minPts = 4 and epsilon (ε) = 1.9

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P1	0											
P2	1.41	0										
P3	2.83	1.41	0									
P4	4.24	2.83	1.41	0								
P5	5.66	4.24	2.83	1.41	0							
P6	5.83	4.47	3.16	2.00	1.41	0						
P7	6.40	5.00	3.61	2.24	1.00	1.00	0					
P8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
P9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
P10	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0		
P11	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
P12	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

P1: P2, P10
P2: P1, P3, P11
P3: P2, P4
P4: P3, P5
P5: P4, P6, P7, P8
P6: P5, P7
P7: P5, P6
P8: P5
P9: P12
P10: P1, P11
P11: P2, P10, P12
P12: P9, P11

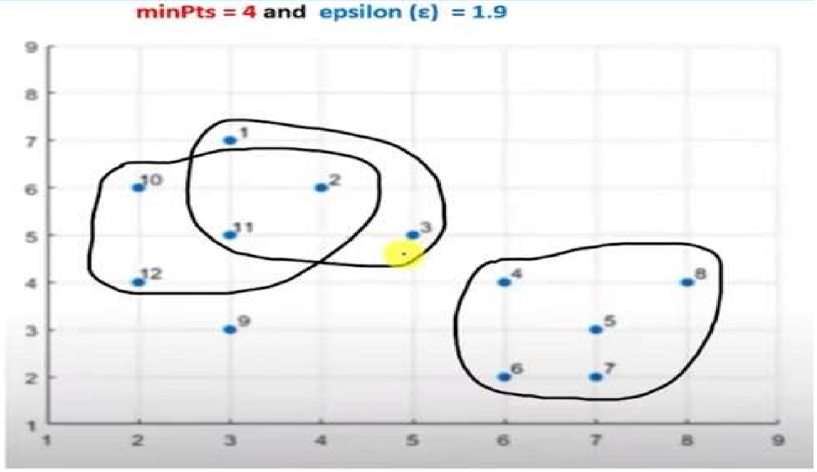
P1: P2, P10
P2: P1, P3, P11
P3: P2, P4
P4: P3, P5
P5: P4, P6, P7, P8
P6: P5, P7
P7: P5, P6
P8: P5
P9: P12
P10: P1, P11
P11: P2, P10, P12
P12: P9, P11

minPts = 4 and epsilon (ε) = 1.9

Point	Status	
P1	Noise	Border
P2	Core	
P3	Noise	Border
P4	Noise	Border
P5	Core	
P6	Noise	Border
P7	Noise	Border
P8	Noise	Border
P9	Noise	
P10	Noise	Border
P11	Core	
P12	Noise	Border

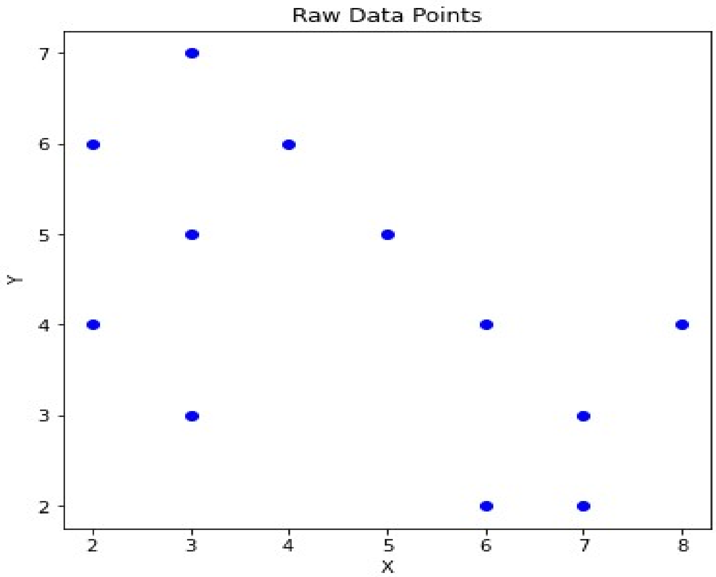
P1: P2, P10
P2: P1, P3, P11
P3: P2, P4
P4: P3, P5
P5: P4, P6, P7, P8
P6: P5, P7
P7: P5, P6
P8: P5
P9: P12
P10: P1, P11
P11: P2, P10, P12
P12: P9, P11

minPts = 4 and epsilon (ε) = 1.9

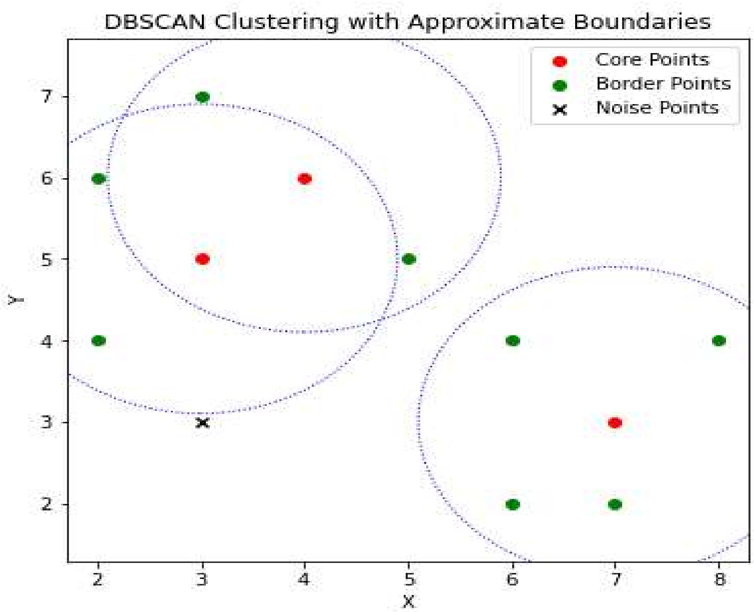


```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import DBSCAN

# Define the data points
points = np.array([ [3, 7], [4, 6], [5, 5], [6, 4], [7, 3], [6, 2], [7, 2], [8, 4], [3, 3], [2, 6], [3, 5], [2, 4]])

# Plot the data points
plt.figure(figsize=(6, 6))
plt.scatter(points[:, 0], points[:, 1], color='b')
plt.title("Raw Data Points")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

# Apply DBSCAN
db = DBSCAN(eps=1.9, min_samples=4).fit(points)
labels = db.labels_

# Extract core points, border points, and noise points
core_points_mask = np.zeros_like(labels, dtype=bool)
core_points_mask[db.core_sample_indices_] = True
border_points_mask = (labels != -1) & ~core_points_mask
noise_points_mask = (labels == -1)

# Plot core points, border points, and noise points
plt.figure(figsize=(6, 6))
plt.scatter(points[core_points_mask, 0], points[core_points_mask, 1], color='red', marker='o', label='Core Points')
plt.scatter(points[border_points_mask, 0], points[border_points_mask, 1], color='green', marker='o', label='Border Points')
plt.scatter(points[noise_points_mask, 0], points[noise_points_mask, 1], color='black', marker='x', label='Noise Points')

# Plot epsilon neighborhoods around core points as circles
for point in points[core_points_mask]:
   circle = plt.Circle((point[0], point[1]), 1.9, color='blue', fill=False, linestyle='dotted')
   plt.gca().add_artist(circle)

plt.title("DBSCAN Clustering with Approximate Boundaries")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.axis('equal')  # Ensure equal scaling of x and y axes
plt.show()
print("DBSCAN Labels:", labels)   # Print the labels assigned by DBSCAN
```

OUTPUT:



Raw Data Points
7
6
5
4
3
2
Y
2
3
4
5
6
7
8
X



DBSCAN Clustering with Approximate Boundaries
Core Points
Border Points
Noise Points
7
6
5
4
3
2
Y
2
3
4
5
6
7
8
X