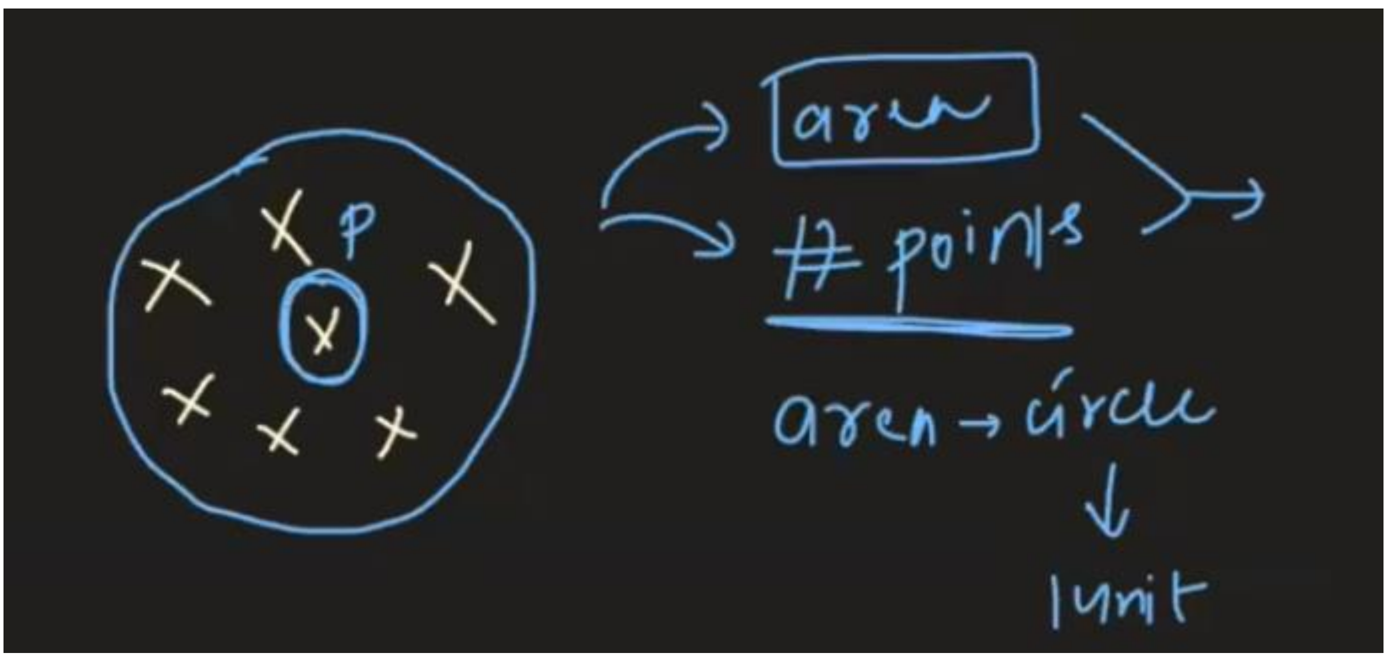
Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Density based clustering algorithms divides your entire dataset into dense regions separated by sparse regions.

MinPts and Epsilon :

How to measure density around a point ?

Measuring density around a point is straightforward — **we define a region around the point and assess the number of points within that designated area.**



To determine density around a point, we employ circles in 2-D, spheres in 3-D, and hyper-spheres in n-dimensional spaces. Suppose we draw unit radius circle around a point P as shown in above figure and here we establish a criterion: a region is considered sparse if it contains fewer than 3 points and dense if it contains 3 or more points.

MinPts stands for “Minimum Points”, is a parameter that specifies the minimum number of points required to form a dense region, which is consider a cluster.

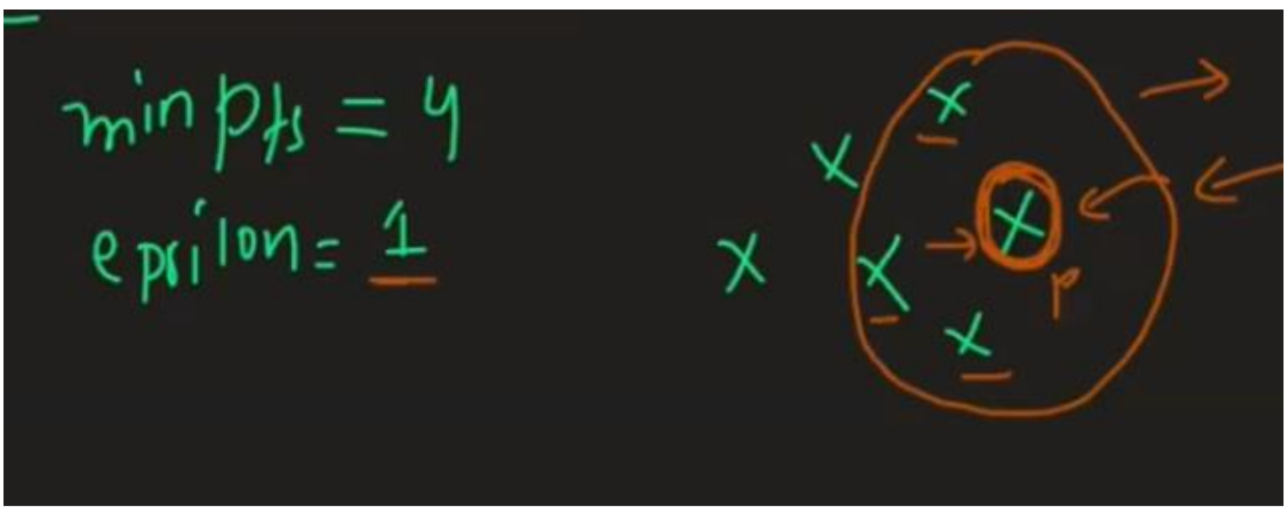
Epsilon is a key parameter that defines the radius of the neighborhood around a given data point.

Specifically, epsilon is the maximum distance between two points for them to be considered as part of the same neighborhood.

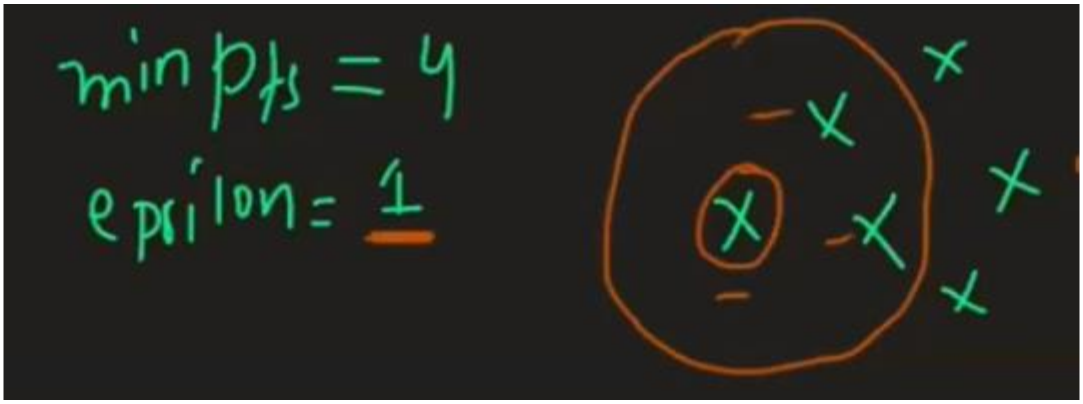
In the above example, the chosen radius value of 1 unit corresponds to epsilon, while the minimum point threshold of 3, determining sparse or dense regions, is representative of MinPts. Both MinPts and Epsilon are hyperparameters that necessitate fine-tuning to achieve optimal results.

Core Points, Border Points and Noise Points :

A point is considered a core point if it has a minimum number of other points(specified by MinPts) within a given radius **ε** of itself.



In the depicted diagram, with ε set to 1 and MinPts to 4, let’s focus on a specific point, P. To determine if P qualifies as a core point, we create a circle with a radius of 1 unit around P. Observing the diagram, it’s evident that point P, along with three additional points within the circle, satisfies the MinPts condition. Hence, we can confidently classify point P as a core point.



Examining the diagram, it's evident that within the circle surrounding a specific point, there are only two points in addition to the point itself, totaling three points. This doesn't meet the MinPts requirement of 4, leading us to conclude that it is not a core point.

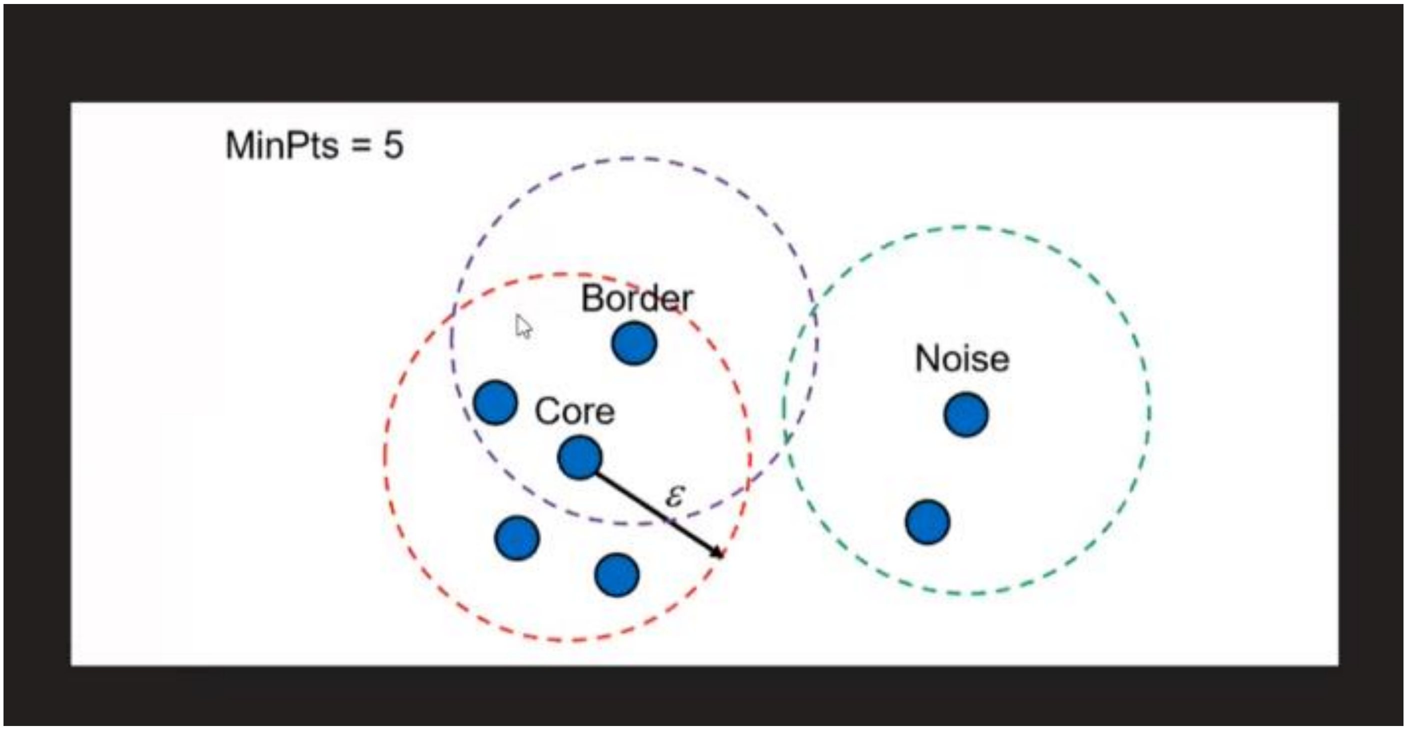
Border Point :

A border point is defined as follows:

- **Not a Core Point:** A border point does not meet the criteria to be a core point. It has fewer than MinPts within its ε-neighbourhood.
- **Neighbor of a Core Point: A border point is within the ε distance of**

  one or more core points. In other words, it lies on the edge of a cluster,

  within the radius ε of at least one core point.



Noise Point :

A noise point is a data point which can neither a core point nor a border point.

Example dataset

point	x	y
p1	4.5	8
p2	5	7
p3	6	6.5
p4	7	5
p5	9	4
p6	7	3
p7	8	3.5
p8	9	5
p9	4	4
p10	3	7.5
p11	4	6
p12	3.5	5

E=1.9 ,min_pt=4			p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
p1	p2,p10	n	0	1.12	2.12	3.91	6.02	5.59	5.7	4.24	4.03	1.58	2.06	3.16
p2	p1,p3,p11	**C**	1.12	0	1.12	2.83	5	4.47	4.61	2.83	3.16	2.06	1.41	2.5
p3	p2,p4	n	2.12	1.12	0	1.8	3.91	3.64	3.61	2.12	3.2	3.16	2.06	2.92
p4	p3,p7	n	3.91	2.83	1.8	0	2.24	2	1.8	0	3.16	4.72	3.16	3.5
p5	p7,p8	n	6.02	5	3.91	2.24	0	2.24	1.12	1.41	5	6.95	5.39	5.59
p6	p7	n	5.59	4.47	3.64	2	2.24	0	1.12	2.83	3.16	6.02	4.24	4.03
p7	p4,p5,p6,p8	**C**	5.7	4.61	3.61	1.8	1.12	1.12	0	2.12	4.03	6.4	4.72	4.74
p8	p5,p7	n	5.41	4.47	3.35	2	1	2.83	1.8	0	5.1	6.5	5.1	5.5
p9	p12	n	4.03	3.16	3.2	3.16	5	3.16	4.03	1.41	0	3.64	2	1.12
p10	p1,p11	n	1.58	2.06	3.16	4.72	6.95	6.02	6.4	3.54	3.64	0	1.8	2.55
p11	p2,p10,p12	**C**	2.06	1.41	2.06	3.16	5.39	4.24	4.72	1.41	2	1.8	0	1.12
p12	p9,p11	n	3.16	2.5	2.92	3.5	5.59	4.03	4.74	0	1.12	2.55	1.12	0

point	checking	
p1	Noise	Border
p2	Cluster	Cluster
p3	Noise	Border
p4	Noise	Border
p5	Noise	Border
p6	Noise	Border
p7	Cluster	Cluster
p8	Noise	Border
p9	Noise	Noise
p10	Noise	Border
p11	Cluster	Cluster
p12	Noise	Border

Python Example

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
```

```python
data = {
    "Cafe": ["p1", "p2", "p3", "p4", "p5", "p6", "p7", "p8", "p9", "p10", "p11",
"p12"],
    "Longitude": [4.5, 5, 6, 7, 9, 7, 8, 9, 4, 3, 4, 3.5],
    "Latitude": [8, 7, 6.5, 5, 4, 3, 3.5, 5, 4, 7.5, 6, 5]
}
```

```
df = pd.DataFrame(data)

df
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[['Longitude', 'Latitude']])
X_scaled
```

```
# 3. Apply DBSCAN
dbscan = DBSCAN(eps=1.0, min_samples=3)
df['Cluster'] = dbscan.fit_predict(X_scaled)

df
```

```
plt.figure(figsize=(8, 6))
for cluster in sorted(df['Cluster'].unique()):
    cluster_data = df[df['Cluster'] == cluster]
    label = f'Cluster {cluster}' if cluster != -1 else 'Noise'
    plt.scatter(cluster_data['Longitude'], cluster_data['Latitude'], s=100,
label=label)

plt.title("DBSCAN Clustering (eps=1.0, min_samples=3)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

In **unsupervised learning** like clustering (e.g., K-Means or DBSCAN), we don't have labels, so we evaluate cluster quality using **internal validation metrics**.

Silhouette Score (Best for Most Use-Cases)

What it Measures:

- How **close each point is to its own cluster** (cohesion)
- Versus how far it is from other clusters (separation)

Formula:

$$s = \frac{b - a}{\max(a, b)}$$

Where:

- a = average distance to points in the **same cluster**
- b = average distance to points in the **nearest other cluster**

Range:

- +1 = perfect clusters
- 0.5 – 0.7 = good
- 0.3 – 0.5 = okay
- < 0.3 = poor

```python
from sklearn.metrics import silhouette_score

score = silhouette_score(X_scaled, df['Cluster'])
print(f"Silhouette Score: {score:.2f}")
```