K-Means Clustering
Color Quantization

- Unsupervised Learning provides opportunities for very creative use cases on algorithm applications.
- Searching for insights, patterns, and general understanding of our data allows us to apply methods to a variety of tasks.

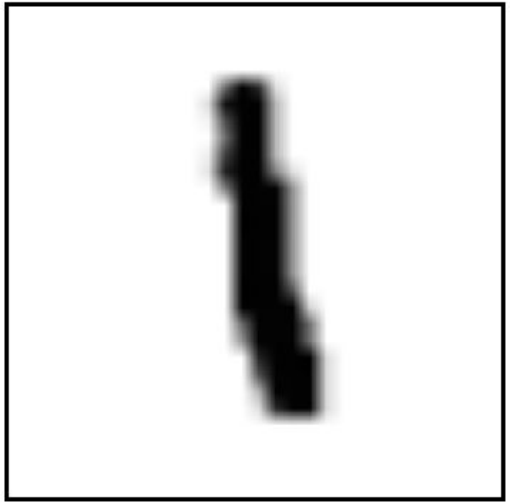
- One application of clustering is on image quantization.
- Let's discuss images, computers, colors, and quantization to get an idea of how K Means clustering can be applied

Imagine an image of a single pen stroke

This image is in **grayscale**. The color range goes from black to white.

You will notice on the edges there are gray colors between black and white.

A computer will store this information as an array with values between a range.



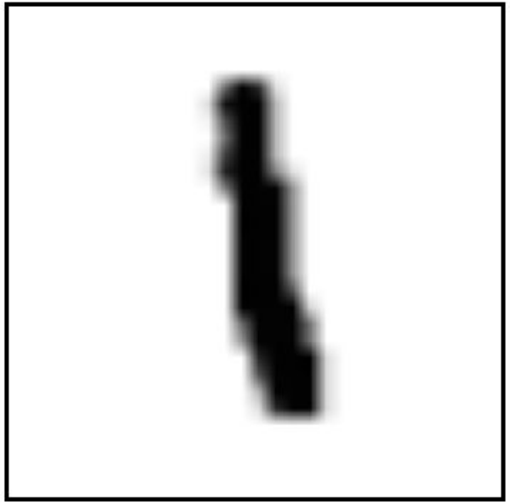
A computer will store this information as an array with values between a range.

Notice 0 is white and 1 is black, with values in between representing gray.

It is also very common for computers to store values from 0-255 for scales.

The range 0 to 255 has to do with how computers store 8-bit numbers.

But you can always divide all the values by 255 to normalize to between 0 and 1

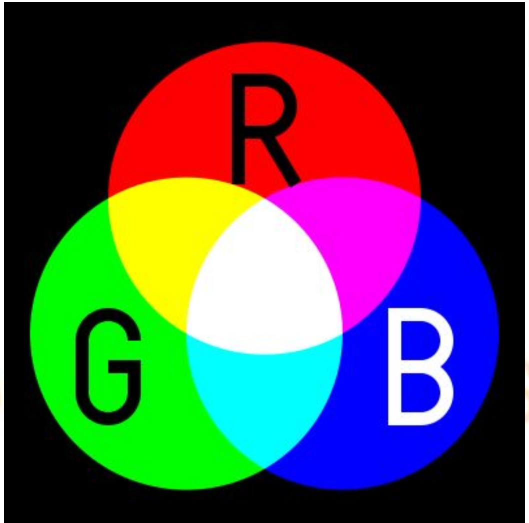


$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

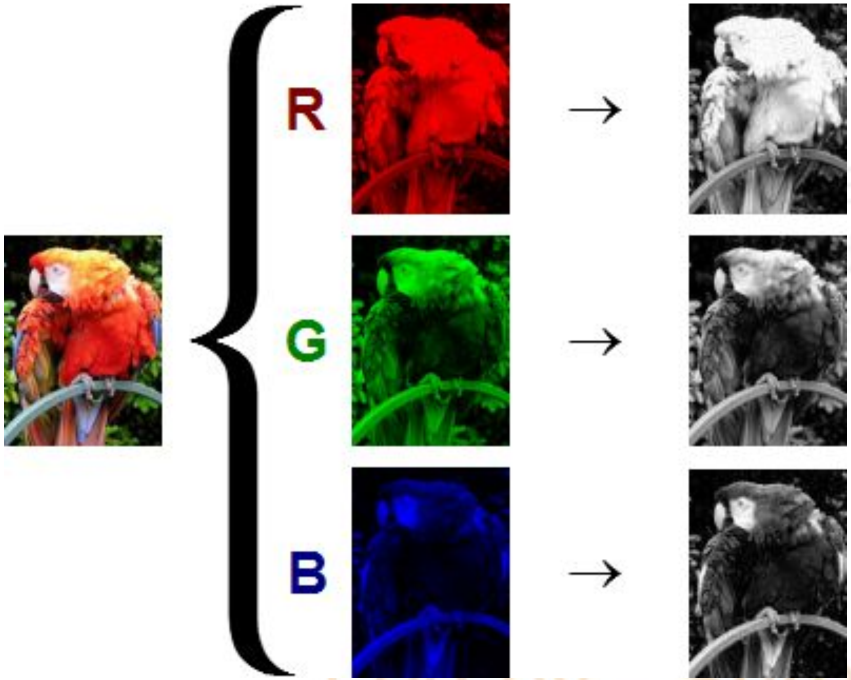
- What about color images?

Color images can be represented as a combination of Red, Green, and Blue.

Additive color mixing allows a wide variety of colors by simply combining different amounts of Red, Green, and Blue.



- RGB can produce a range of colors
- Each color channel will have intensity values.
- You may have already seen this sort of representation in other software with RGB sliders.
- Notice we now have 3 distinct values to track, with each value in a range (shown here from 0-255).
- Combining RGB to produce a distinct color.
- From a computer perspective, this looks like 3 arrays, each array representing a color channel.
- For eg, a single pixel (1 by 1 image) here is (213,111,56) for (R,G,B).
- But How is this stored for a larger color image?



- The shape of the color array then has 3 dimensions.

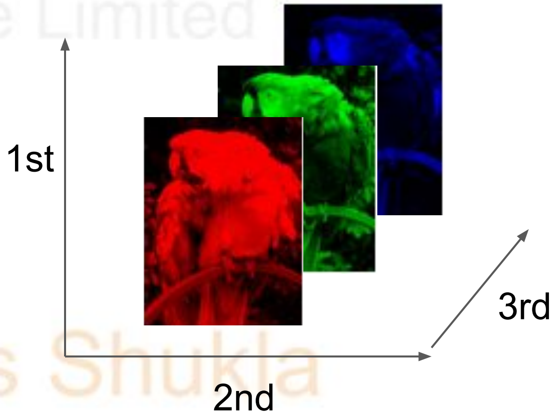
1. Height
2. Width
3. Color Channels

This means when you read in an image and check its shape, it will look something like: **(1280,720,3)**

- **1280** pixel width
- **720** pixel height
- **3** color channels

This means when you read in an image and check its shape, it will look something like:
(720,1280,3)

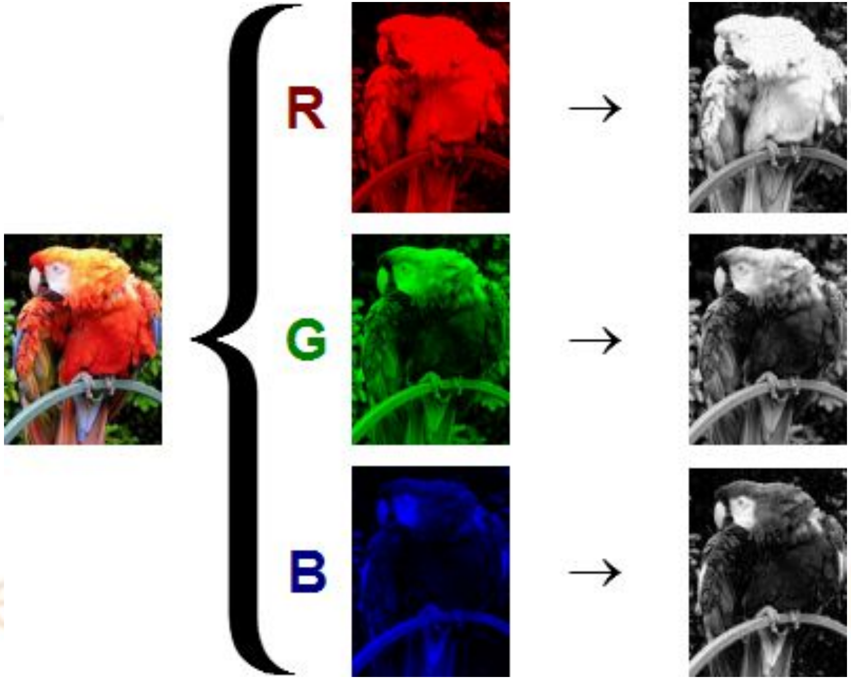
- **720** pixel height
- **1280** pixel width
- **3** color channels



Keep in mind the computer won't "know" a channel is Red, it just knows that there are now 3 intensity channels.

The user needs to dictate which channel is for which color.

Each channel alone is essentially the same as a grayscale image.



From the computer's perspective you simply have an array with 3 dimensions, where a user or display function can attribute each dimension to a color channel (e.g. red intensity).

Swapping these arrays across channels would allow for effects such as color inversion.

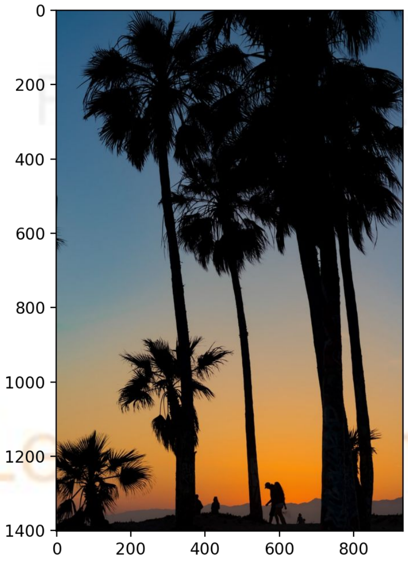
Now how can we apply clustering to RGB color channels and images?

Imagine the following image.

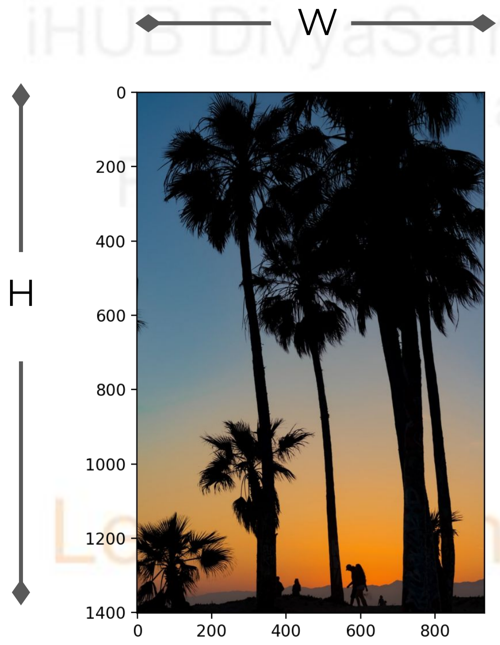
There are many shades of colors in this image, with many (R,G,B) combinations.

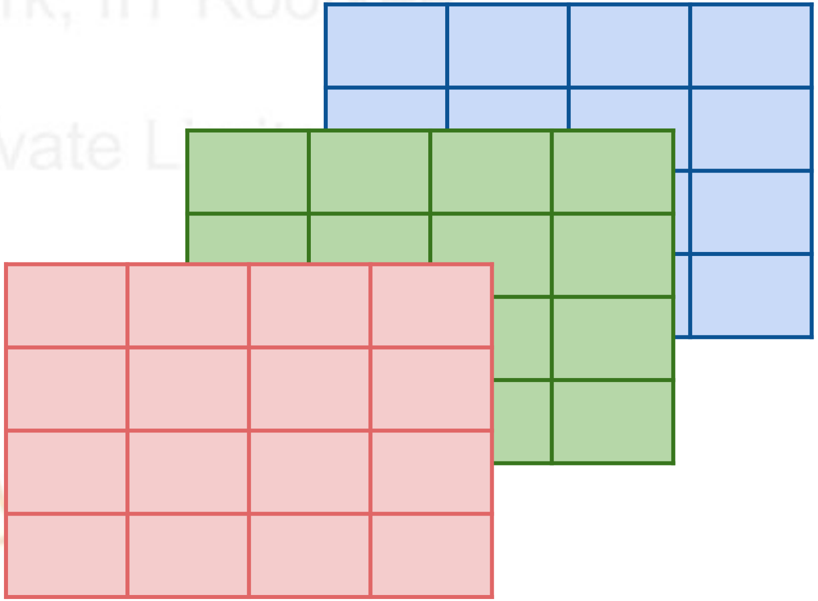
What if we wanted to reduce this to 6 colors for simplified display purposes?

What if we wanted to compress the image for a smaller screen with less colors?

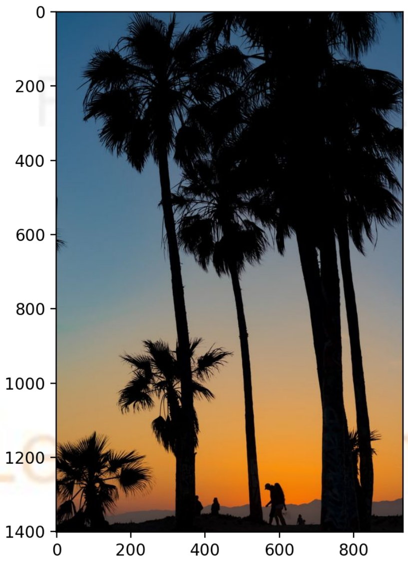


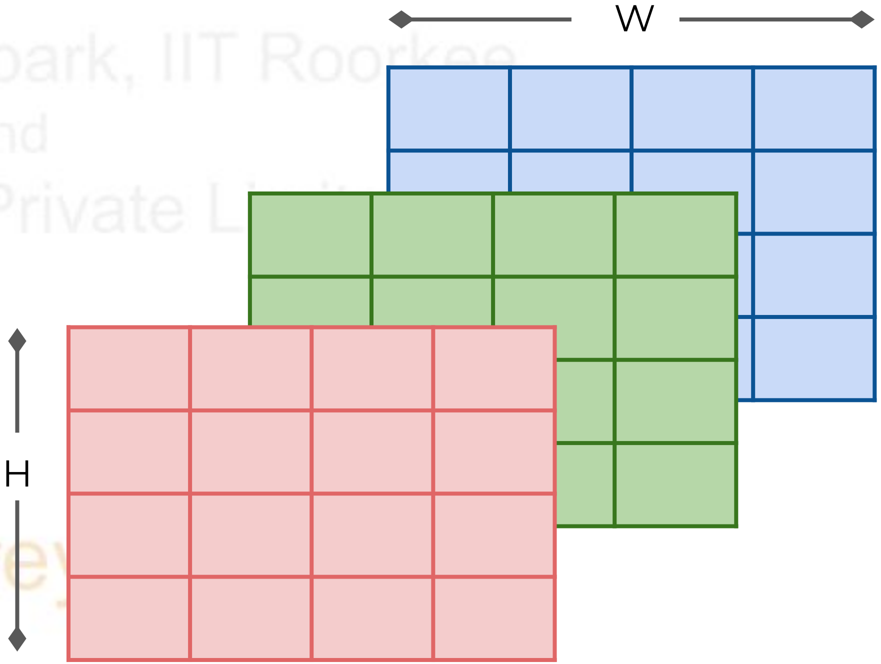
Recall the image is a 3D array (H,W,C):



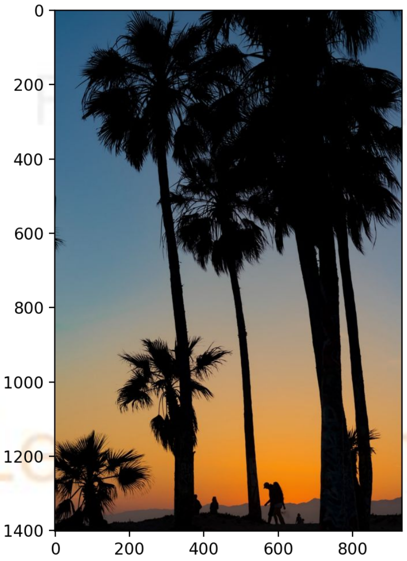


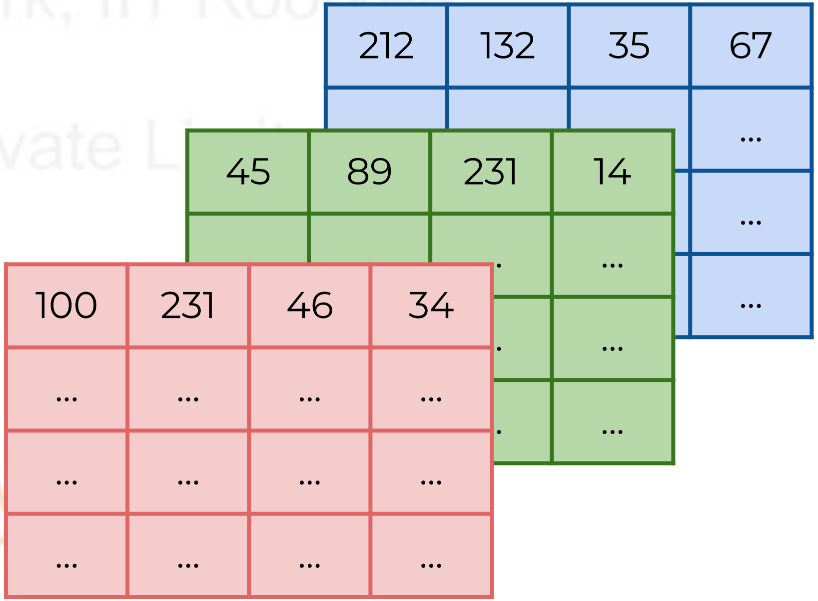
Recall the image is a 3D array (H,W,C):



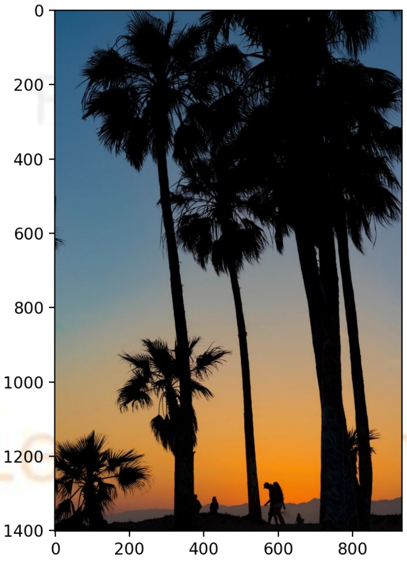


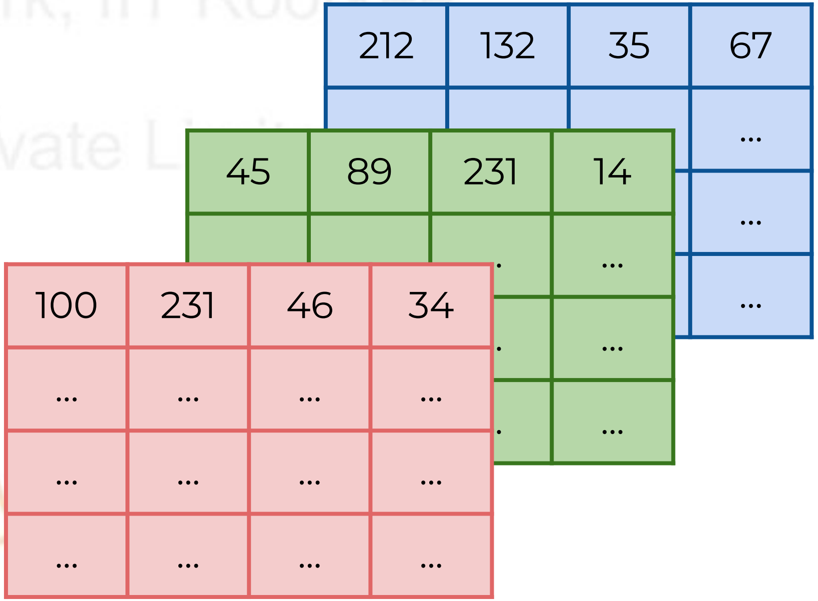
Each pixel has an RGB value to create a color:





We can reshape the image to an X array feature set, with features R,G,B:

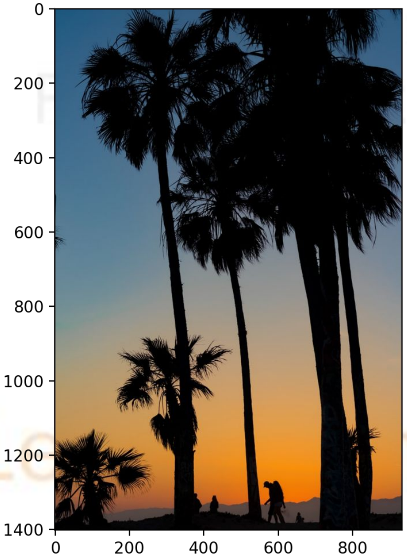




We can reshape the image to an X array feature set, with features R,G,B:

W

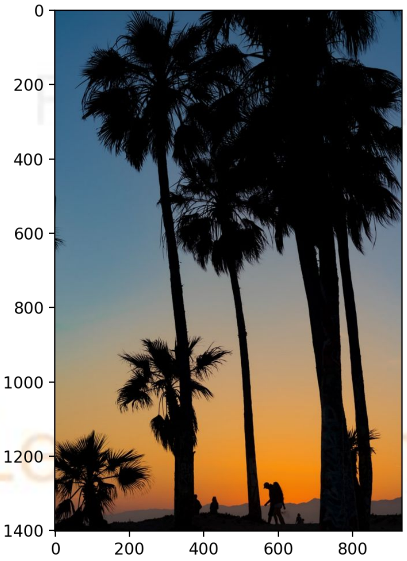
H



H x W

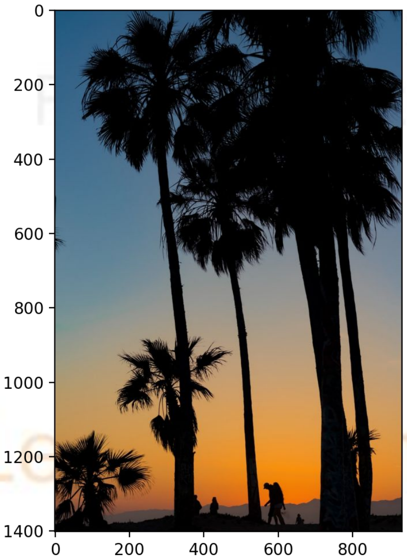
R	G	B
100	15	132
231	23	78
...	...	...
46	243	164
34	145	67

We then choose a K value of colors and use K Means clustering to create labels:



R	G	B
100	15	132
231	23	78
...	...	...
46	243	164
34	145	67

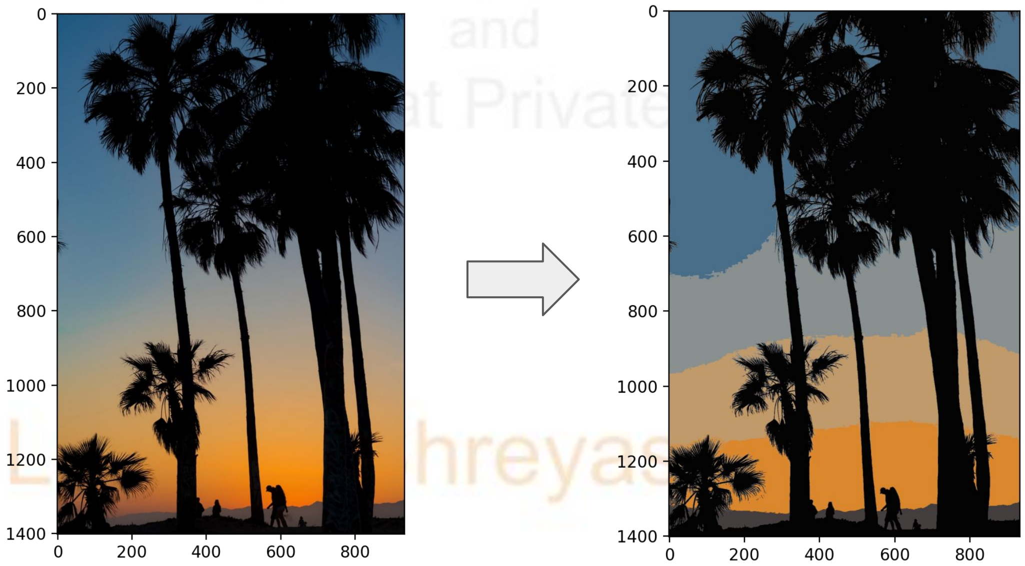
- We then choose a K value of colors and use K Means clustering to create labels.
- Recall each cluster also has a **center** in the N dimensional feature space
- Meaning each cluster center is an average (R,G,B) value we can use for reassignment!



R	G	B	Cluster
100	15	132	0
231	23	78	1
...	...	...	...
46	243	164	2
34	145	67	0

We can then grab each data point and convert it to the same value as the center.

This directly reduces to K color values (known as quantization).



Let’s explore this in practice !!

DBSCAN

DBSCAN

Density-based spatial clustering of applications with noise is a powerful technique which can be used for clustering and outlier detection.

- Intuition of DBSCAN
- DBSCAN vs. K-Means Clustering
- DBSCAN Hyperparameters Theory
- DBSCAN Hyperparameters Coding

Theory and Intuition

DBSCAN stands for **D**ensity-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise.

Some Questions:

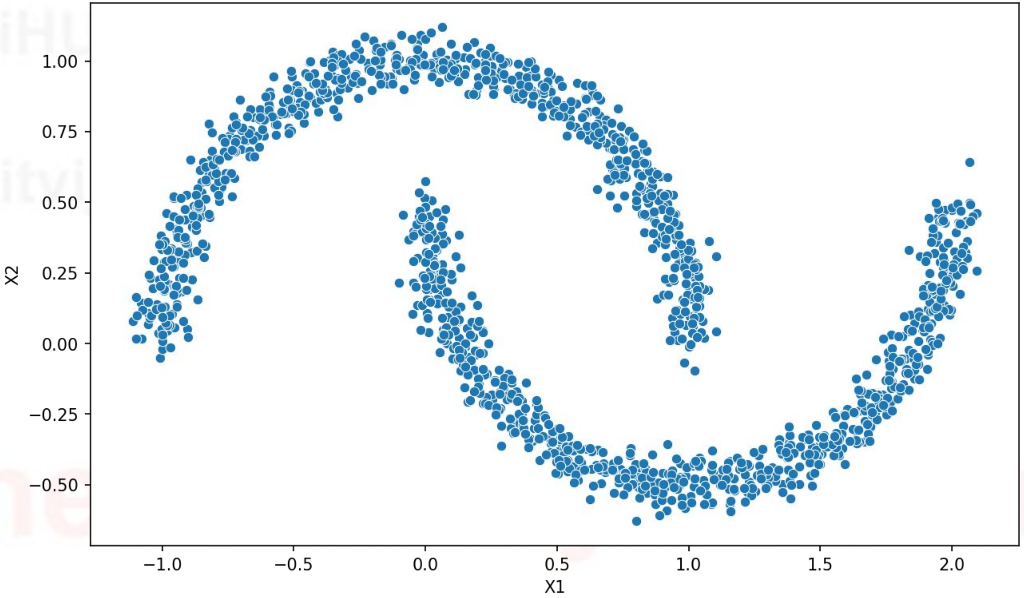
- How does DBSCAN work?
- Advantages and disadvantages of DBSCAN?
- How does it deal with outliers and noise?

Key Ideas

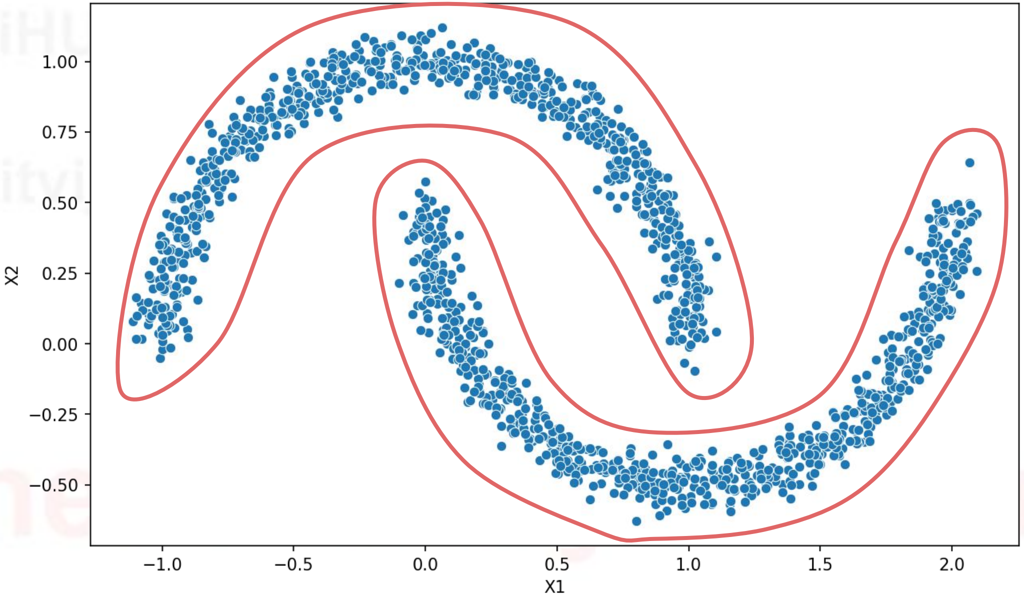
- DBSCAN focuses on using **density** of points as its main factor for assigning cluster labels.

- This creates the ability to find cluster segmentations that other algorithms have difficulty with.

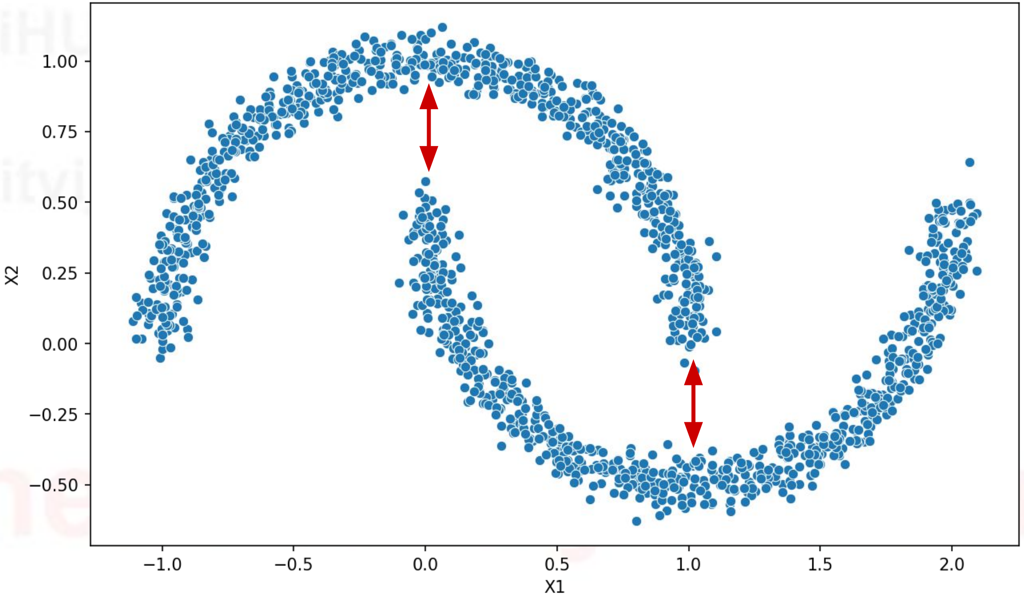
Consider the following data set:



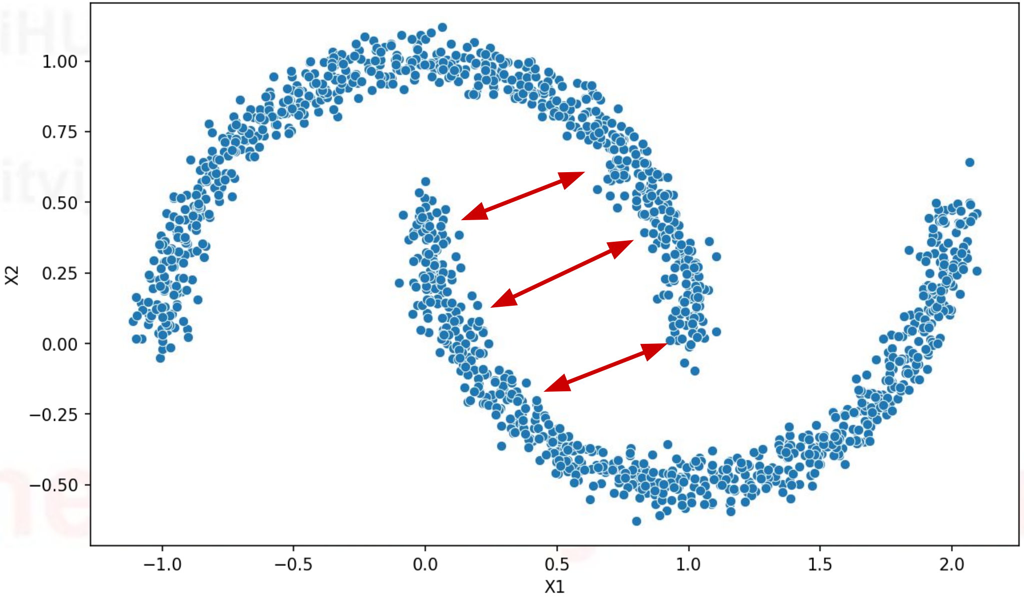
Cleary two “moon” shaped clusters:



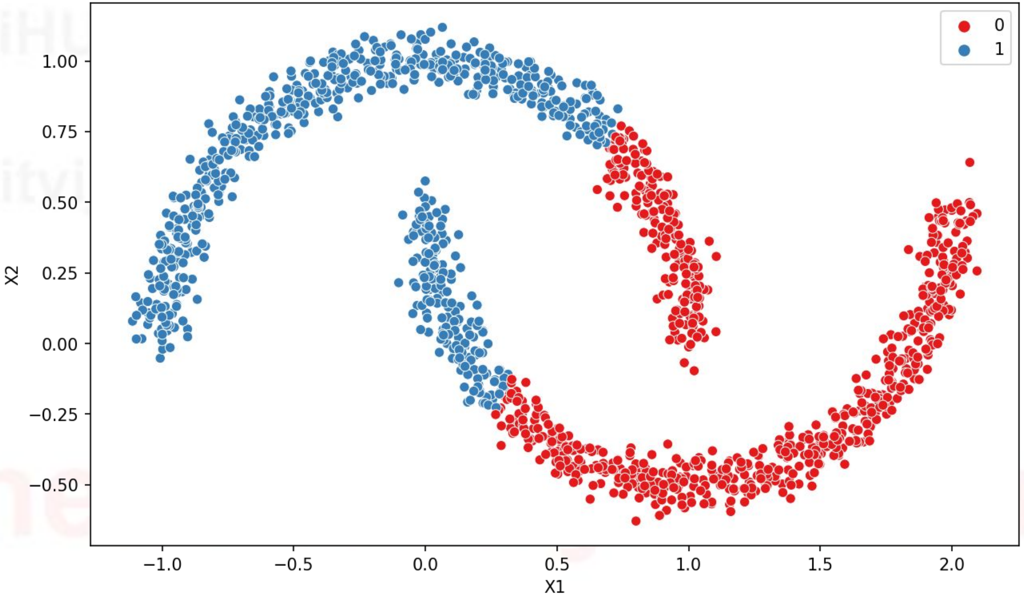
But distance based clustering has issues:



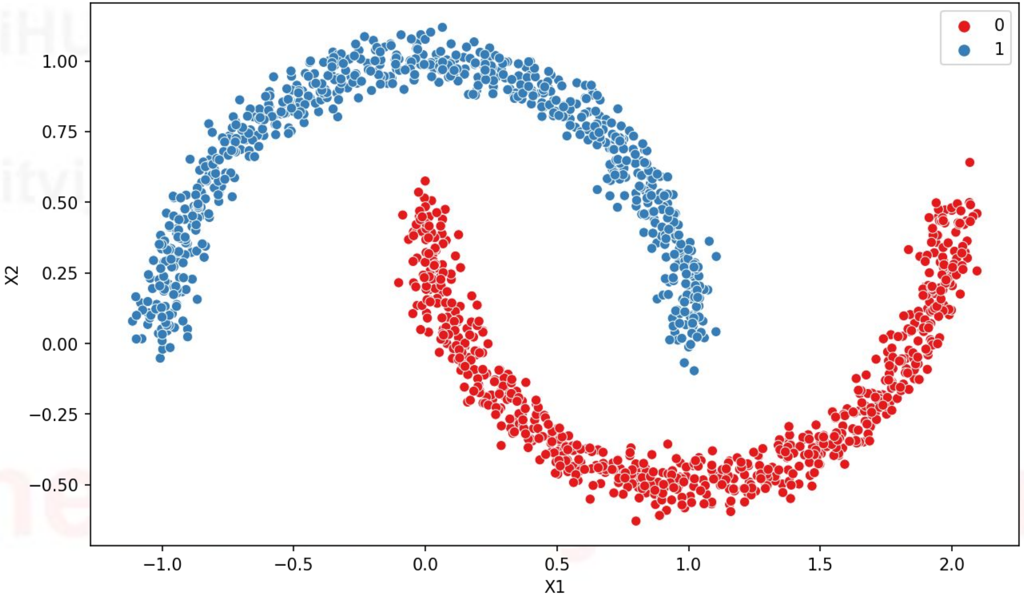
But distance based clustering has issues:



Results of K-Means:



Results of DBSCAN:



DBSCAN iterates through points and uses two key hyperparameters (epsilon and minimum number of points) to assign cluster labels.

Unlike K-Means, it focuses on density as the main factor for cluster assignment of points.

Key Hyperparameters:

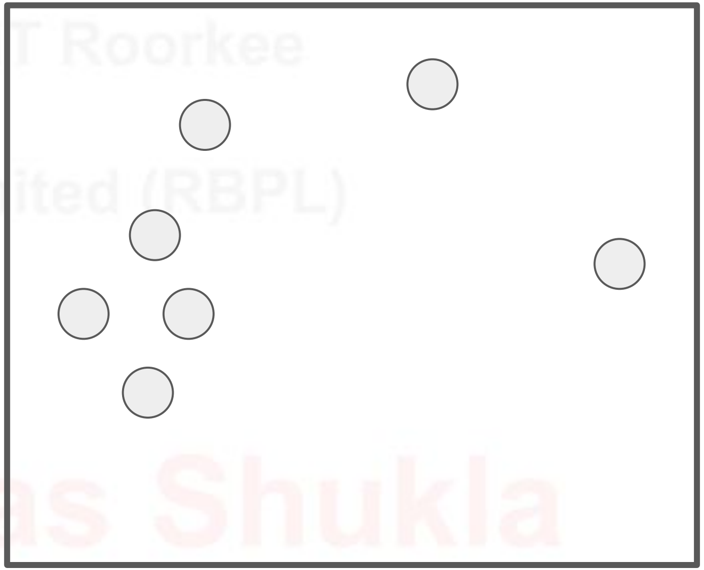
- Epsilon:
  - Distance extended from a point.
- Minimum Number of Points:
  - Minimum number of points in an epsilon distance.

DBSCAN Point Types:

- Core
- Border
- Outlier

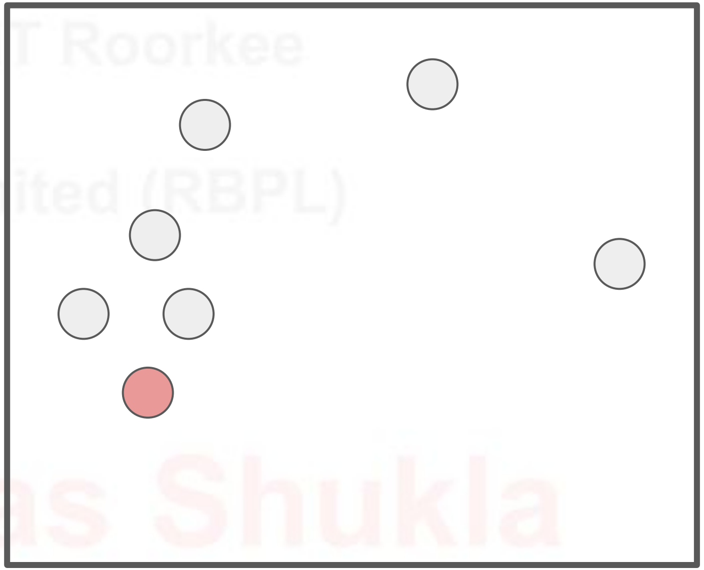
DBSCAN Point Types:

- Core
- Border
- Outlier



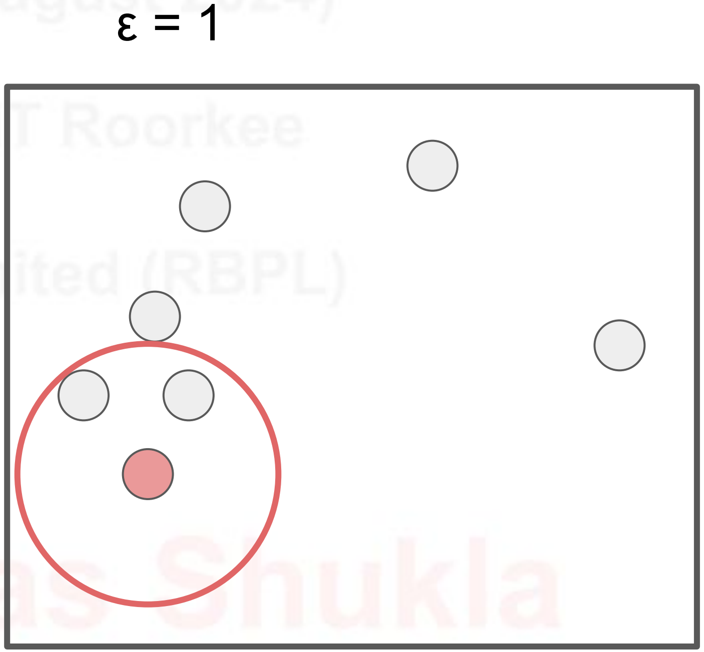
DBSCAN Point Types:

- Core



DBSCAN Point Types:

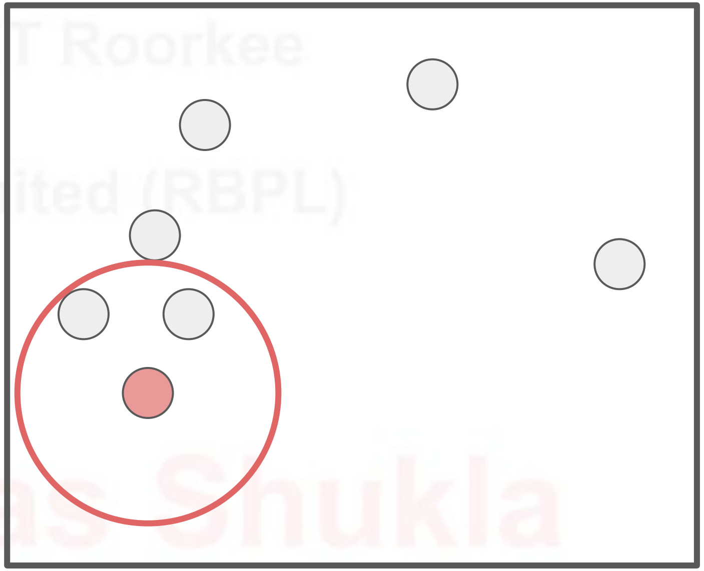
- Core



DBSCAN Point Types:

- Core

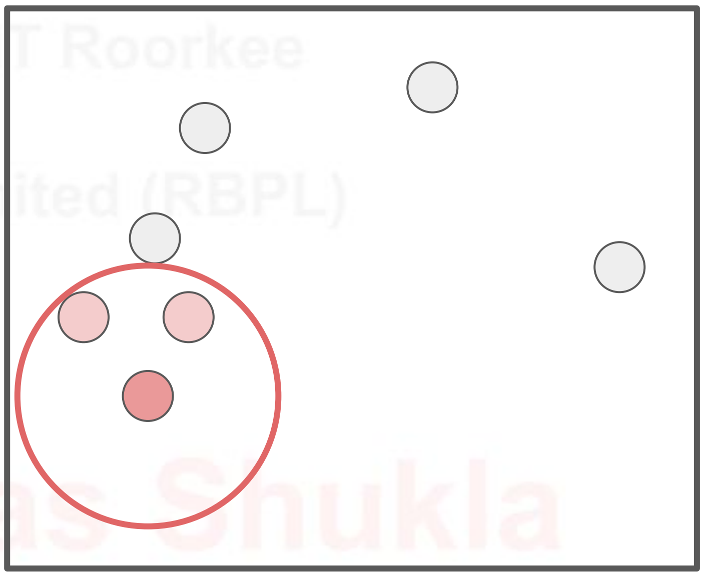
$\varepsilon = 1$ and Min Points = 2



DBSCAN Point Types:

- Core

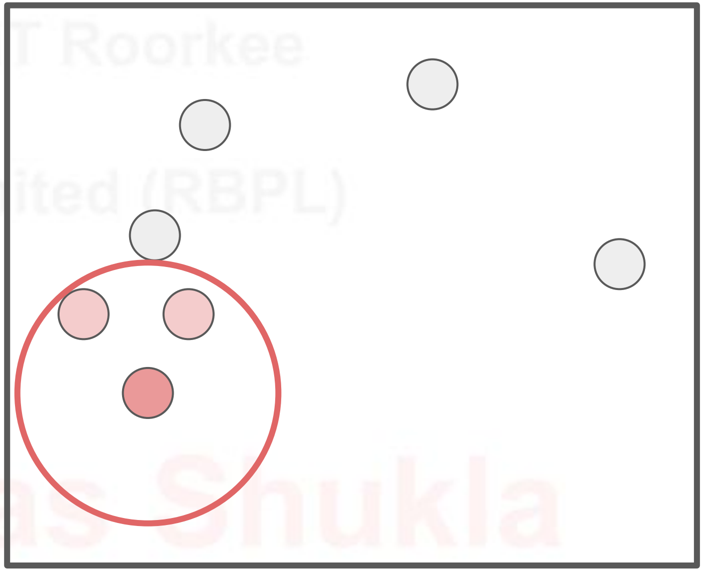
$\varepsilon = 1$ and Min Points = 2



DBSCAN Point Types:

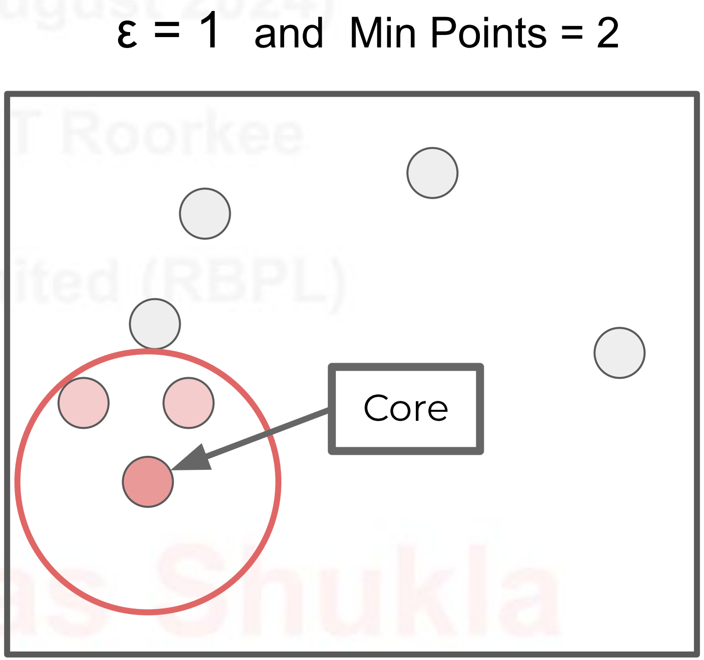
- Core:
  - Point with min. points in epsilon range.

$\varepsilon = 1$ and Min Points = 2



DBSCAN Point Types:

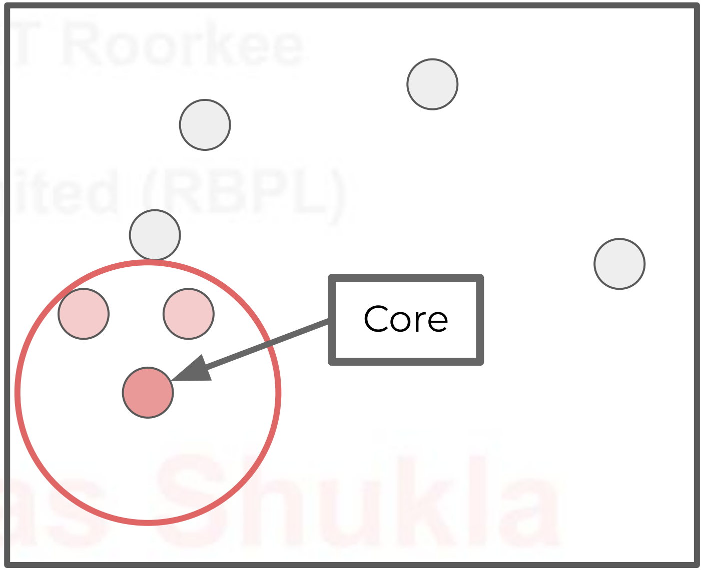
- Core:
  - Point with min. points in epsilon range.



DBSCAN Point Types:

- Core:
  - Point with min. points in epsilon range (including itself).

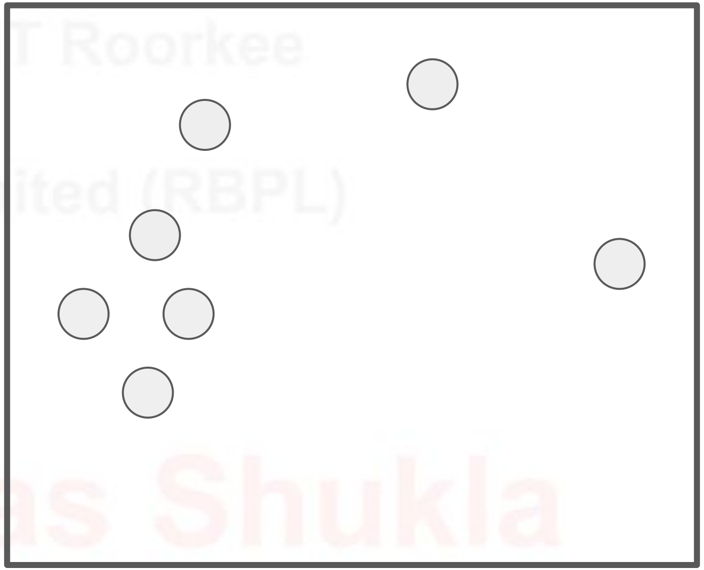
ε = 1 and Min Points = 3



DBSCAN Point Types:

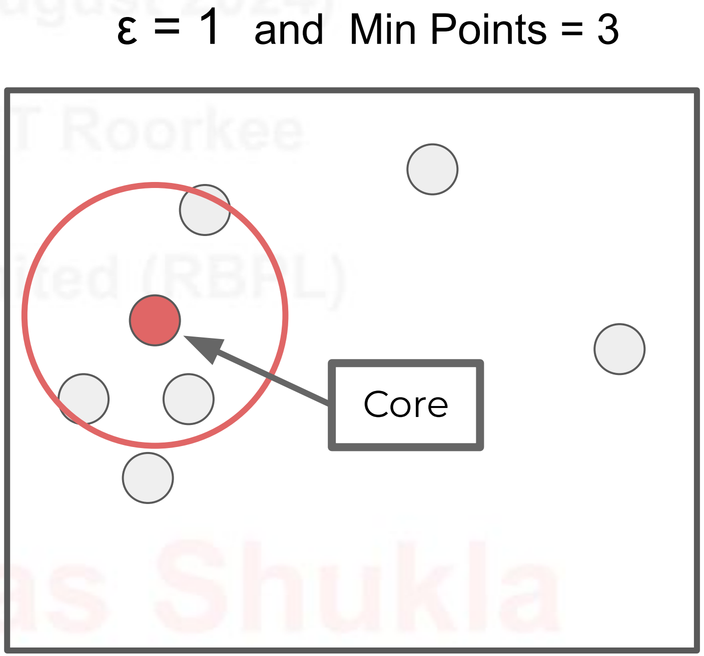
- Border:
  - In epsilon range of core point, but does not contain min. number of points.

ε = 1 and Min Points = 3



DBSCAN Point Types:

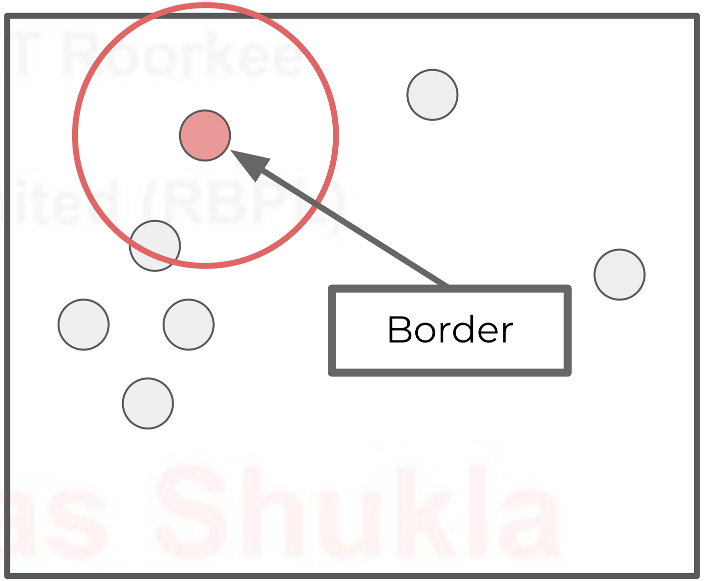
- Border:
  - In epsilon range of core point, but does not contain min. number of points.



DBSCAN Point Types:

- Border:
  - In epsilon range of core point, but does not contain min. number of points.

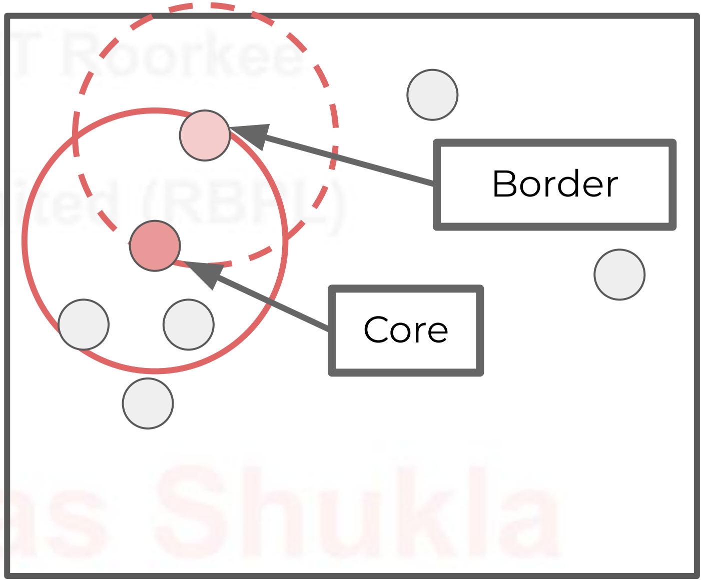
ε = 1 and Min Points = 3



DBSCAN Point Types:

- Border:
  - In epsilon range of core point, but does not contain min. number of points.

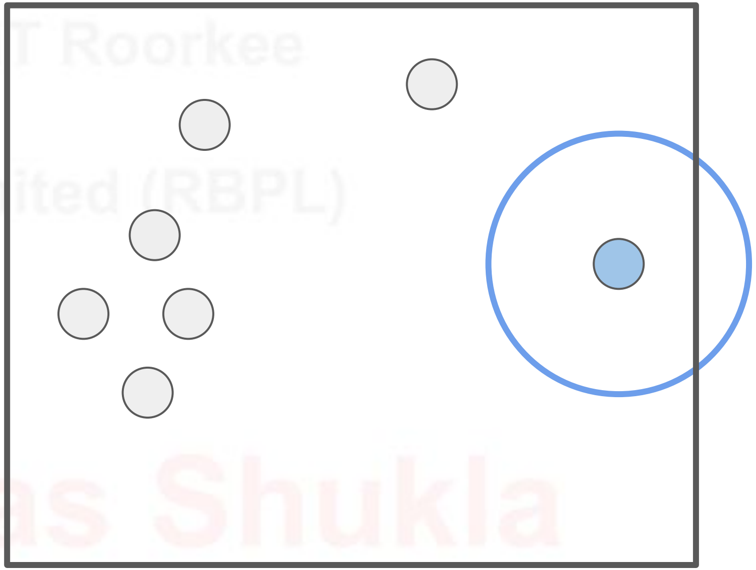
ε = 1 and Min Points = 3



DBSCAN Point Types:

- Outlier:
  - Can not be "reached" by points in a cluster assignment.

ε = 1 and Min Points = 3



Let's review the actual process of DBSCAN for assigning clusters.

DBSCAN Procedure:

- Pick a random point not yet assigned.
- Determine the point type.
- Once a **core** point has been found, add all directly reachable points to the same cluster as core.
- Repeat until all points have been assigned to a cluster or as an outlier.

Coding Example on Data Sets

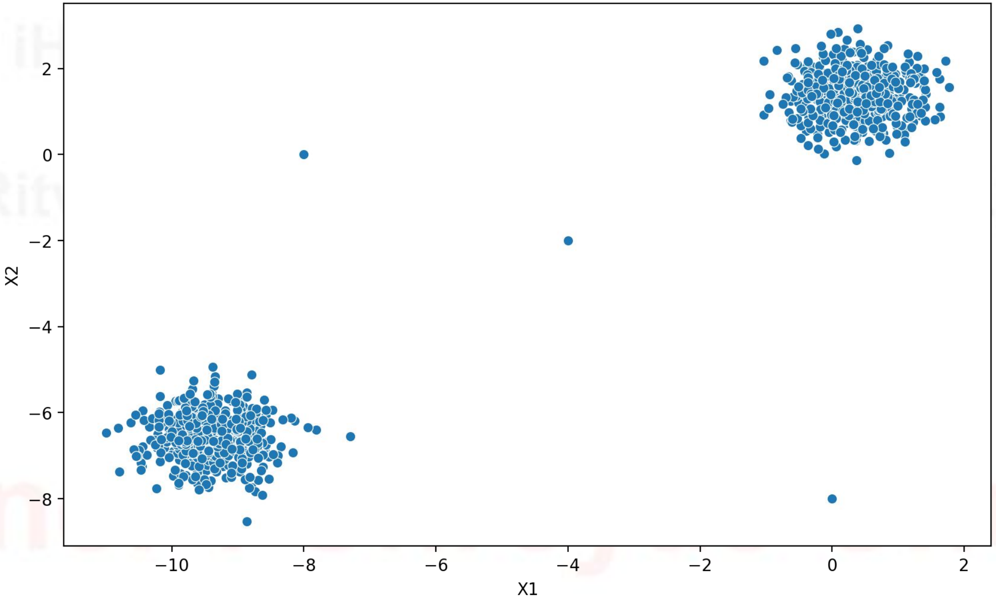
Key Hyperparameters

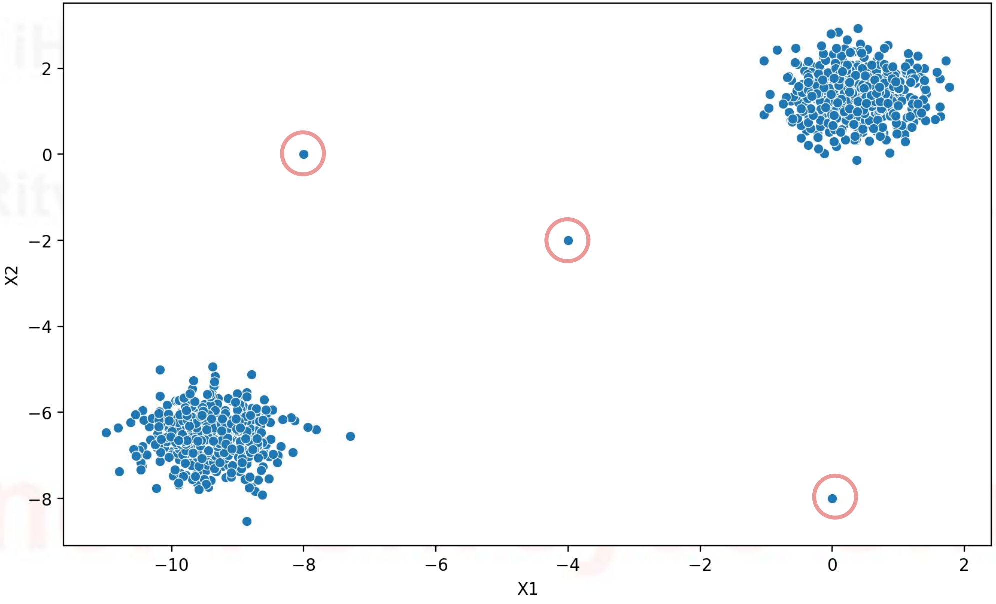
Two key hyperparameters for DBSCAN:

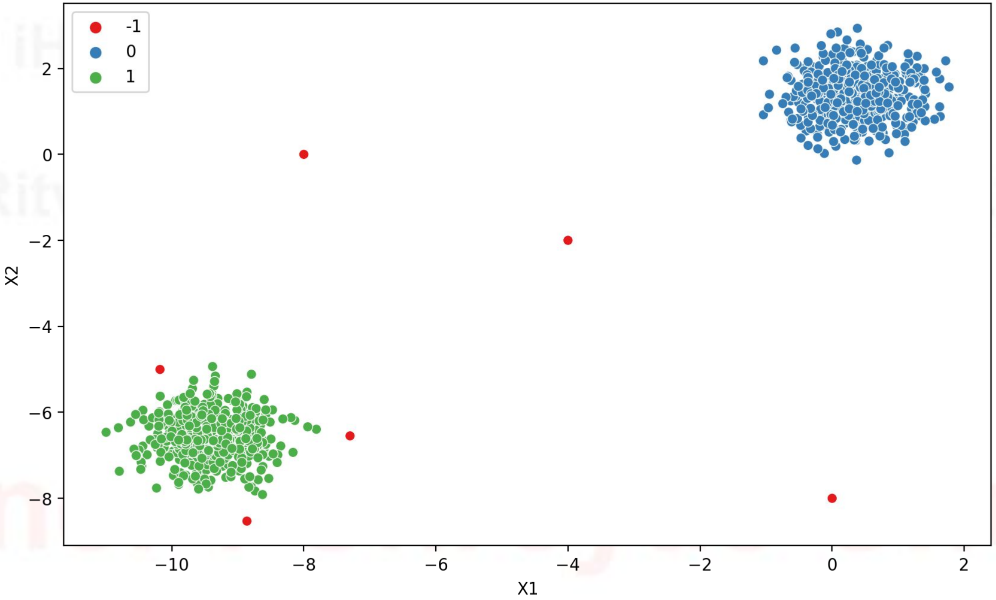
- Epsilon:
  - Distance extended from a point to search for Min. Number of Points.
- Min. Number of Points:
  - Min. Number of Points within Epsilon distance to be a core point.

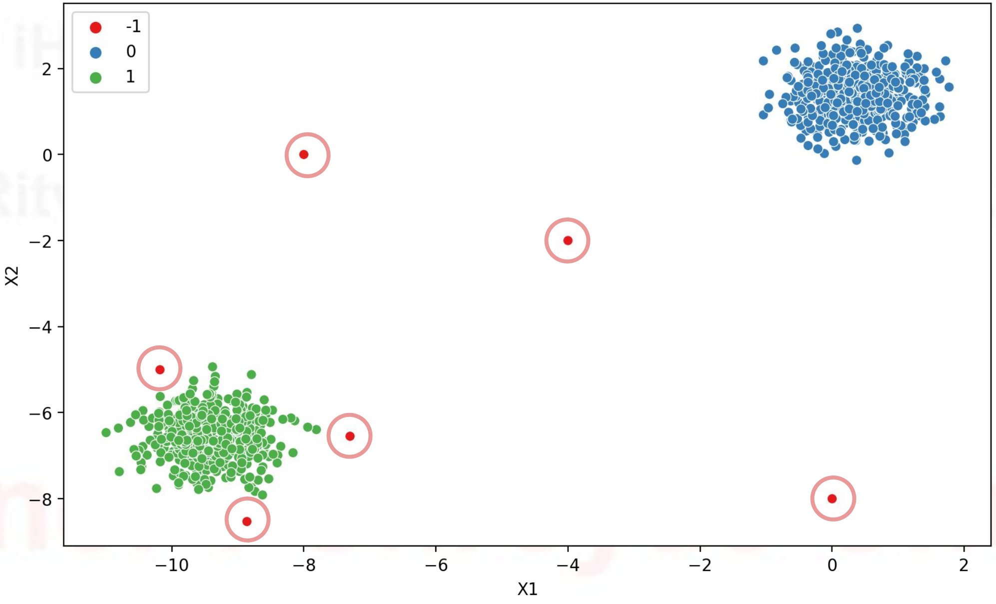
Adjusting these hyperparameters have two main outcomes:

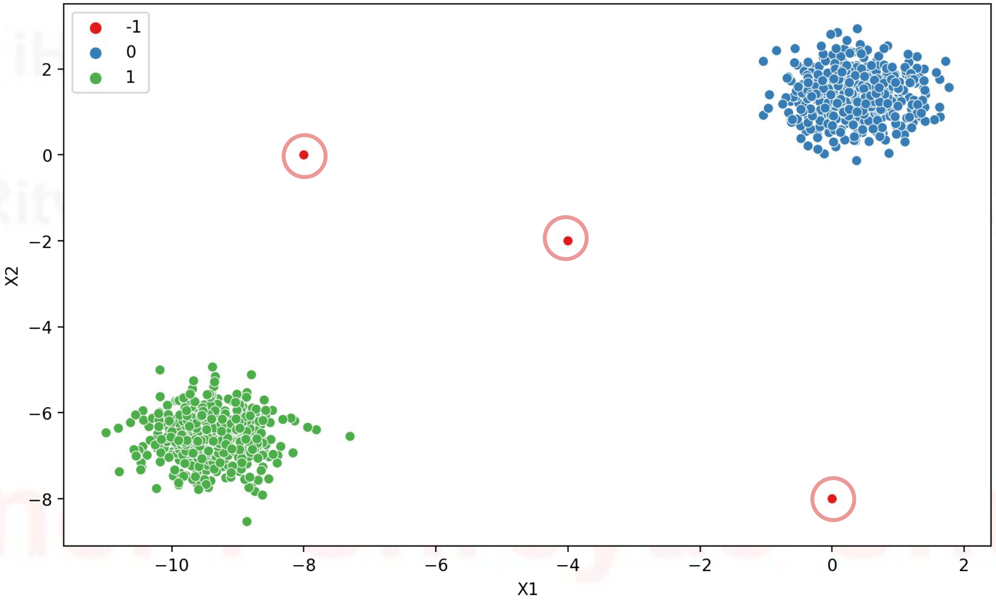
- Changing number of clusters.
- Changing what is an outlier point.


X2
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
X1


X2
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
X1


-1
0
1
X2
X1
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2


-1
0
1
X2
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
X1


-1
0
1
X2
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
X1

Epsilon Intuition:

- Increasing epsilon allows more points to be **core** points which also results in more **border** points and less outlier points.
- Imagine a huge epsilon, all points would be within the neighborhood and classified as the same cluster!
- Decreasing epsilon causes more points not to be in range of each other, creating more unique clusters.
- Imagine a tiny epsilon, the range would not extend far out enough to come into contact with any other points!

Methods for finding an epsilon value:

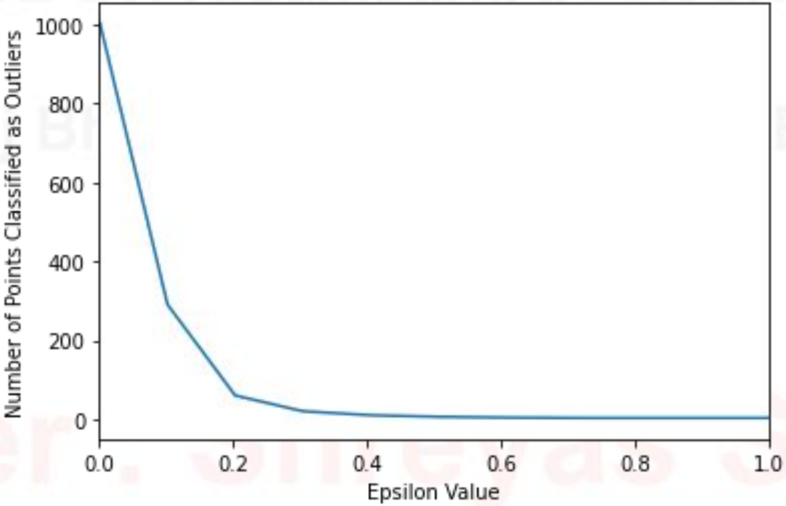
Run multiple DBSCAN models varying epsilon and measure:

- Number of Clusters
- Number of Outliers
- Percentage of Outliers
-

Extremely dependent on the particular data set and domain space.

Requires user to have some expectation or intuition about number of clusters and relative percentage of outliers.

Plot “elbow/knee” diagram comparing epsilon values:



Minimum Number of Samples/Points:

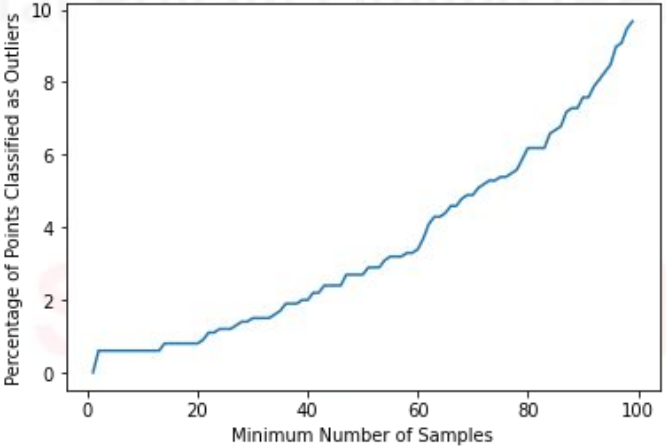
- Number of samples in a neighborhood for a point to be considered as a **core** point (including the point itself).

Min. Number of Samples Intuition:

- Increasing to a larger number of samples needed to be considered a core point, causes more points to be considered unique outliers.
- Imagine if min. number of samples was close to total number of points available, then very likely all points would become outliers.

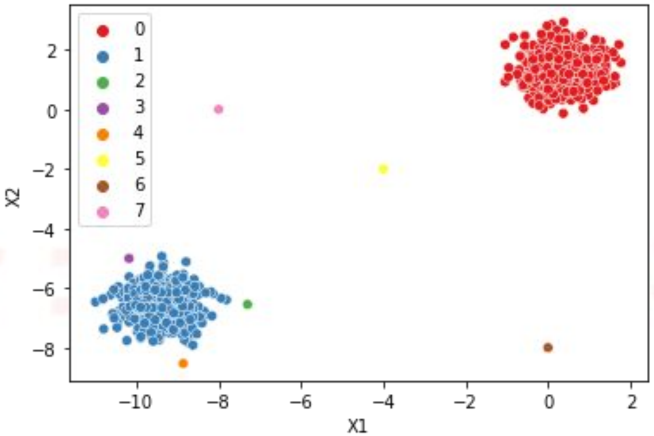
Choosing Min. Number of Samples:

- ○ Test multiple potential values and chart against number of outliers labeled.



Min. Number of Samples Note:

- Useful to increase to create potential new small clusters, instead of complete outliers.



Let’s continue by exploring hyperparameters with code and data examples!