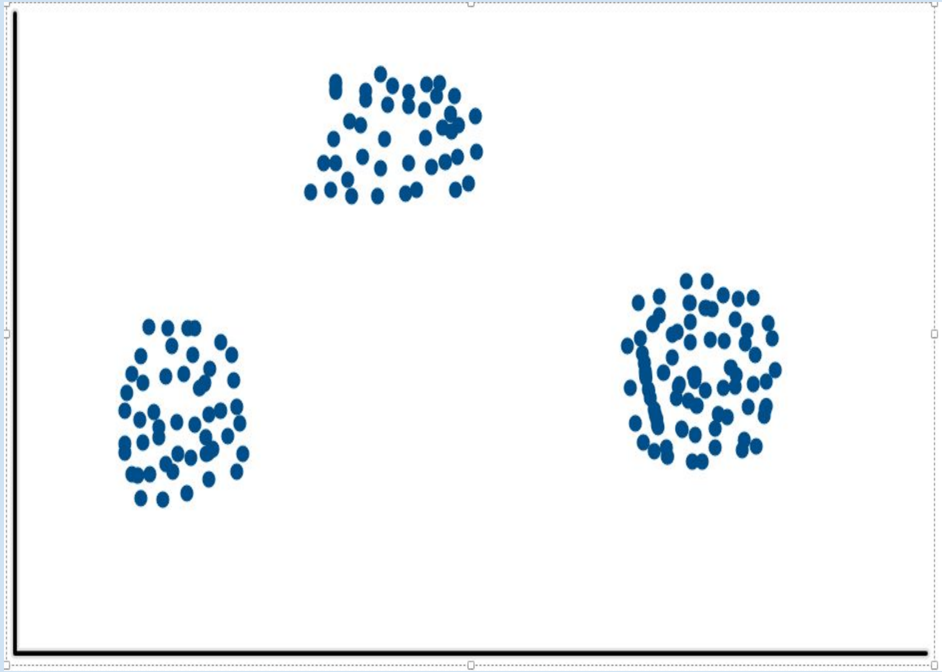
DBSCAN

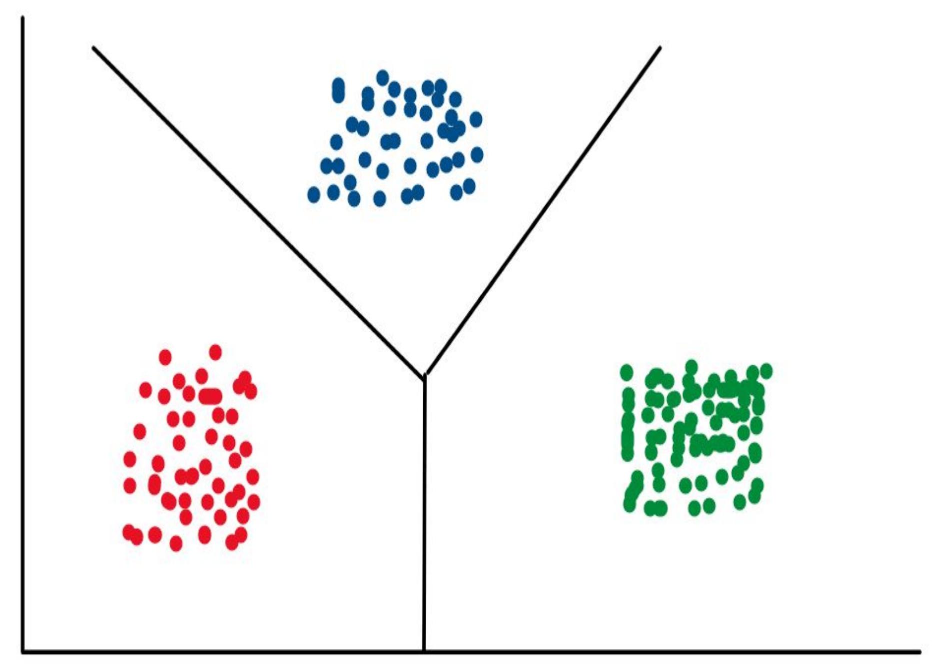
By Group 2: Elsa, Jerad, Ryan, Shannon

Clustering

Original Data



Clustered Data



Overview of DBSCAN

- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Clustering algorithm
- Metric: distance between points
- Clusters:
  - Areas of *high density* separated by areas of *low density*
  - A cluster consists of *core points* (close to each other) and *noncore points* (close to core points)
- Not possible to train/test

...But What IS Density? (Hyperparameters)

Density is defined with two hyperparameters:

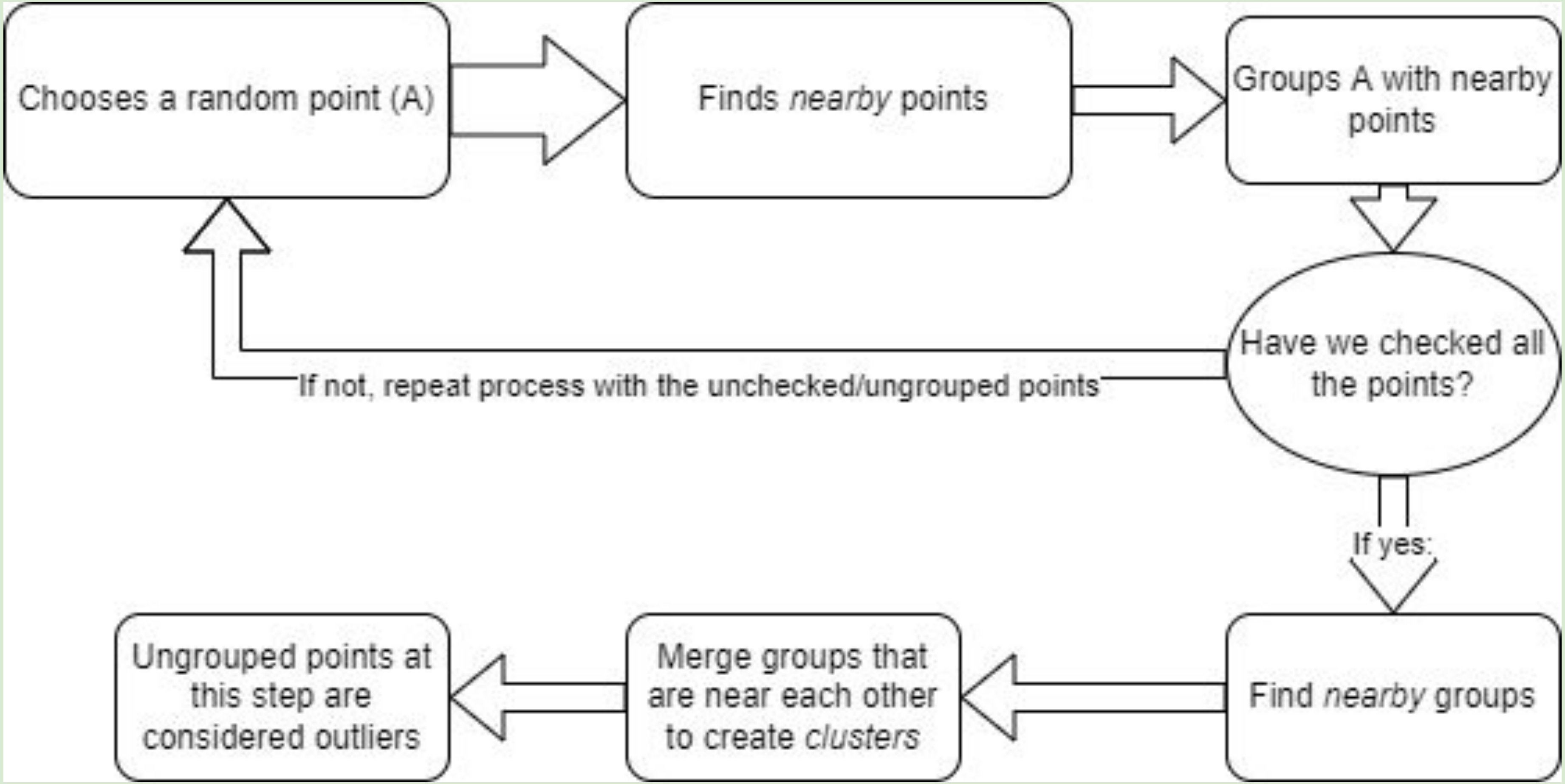
- **eps:** the maximum distance for points to be considered “close”
- **min_samples:** the minimum number of “close” points necessary to create a “group”

Min_samples are required within distance *eps* to form an initial *core group*.

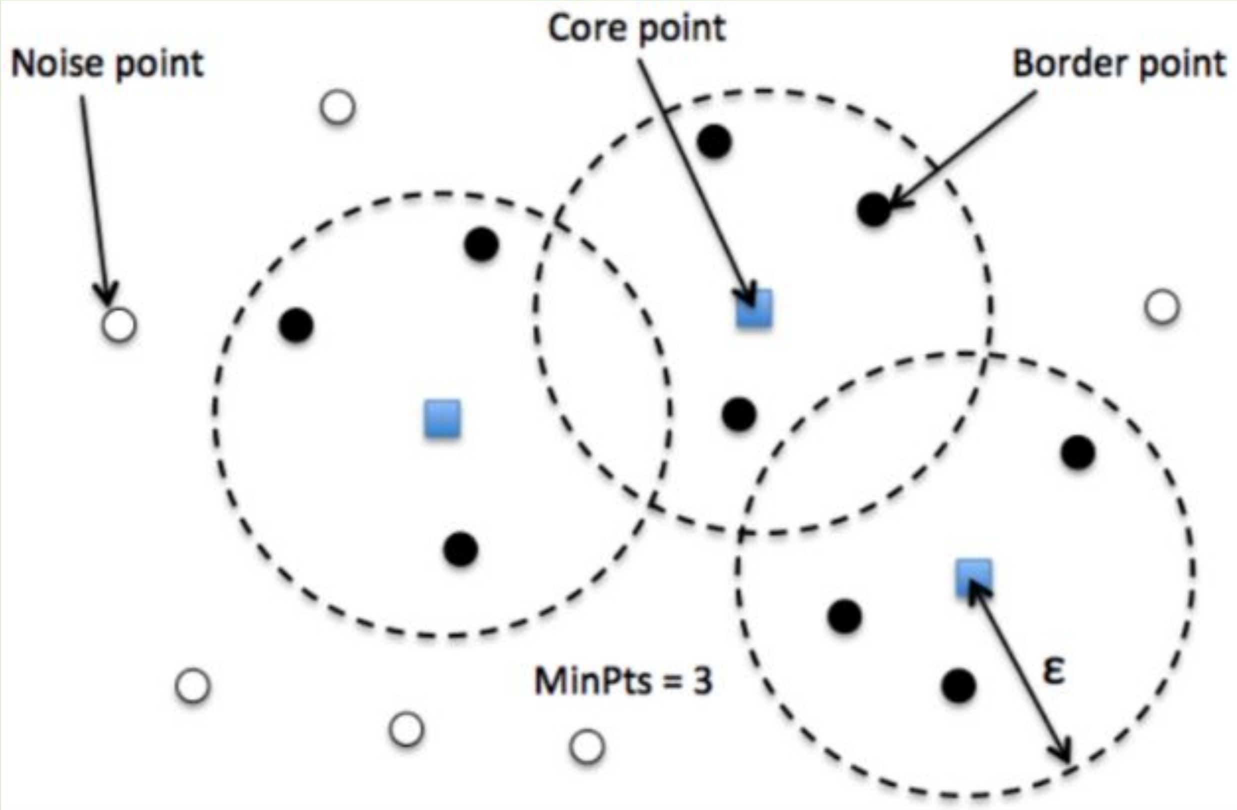
From there, clusters are formed out of core groups.

Definitions paraphrased from https://scikit-learn.org/stable/modules/clustering.html#dbscan

DBSCAN Process



DBSCAN Process

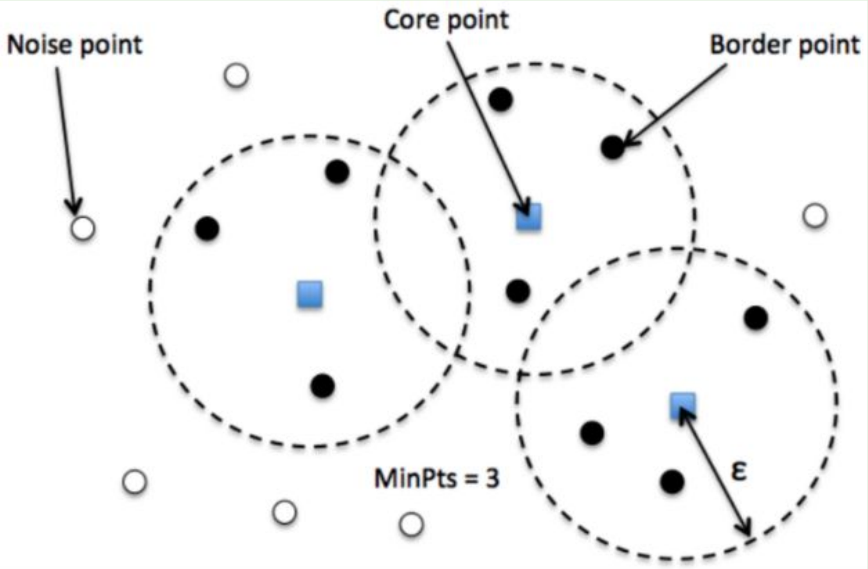


DBSCAN Pseudocode

```
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                  /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)     /* Find neighbors */
        if |N| < minPts then {                              /* Density check */
            label(P) := Noise                               /* Label as Noise */
            continue
        }
        C := C + 1                                          /* next cluster label */
        label(P) := C                                       /* Label initial point */
        SeedSet S := N \ {P}                                /* Neighbors to expand */
        for each point Q in S {                             /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C          /* Change Noise to border point */
            if label(Q) ≠ undefined then continue           /* Previously processed (e.g., border point) */
            label(Q) := C                                   /* Label neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
            if |N| ≥ minPts then {                          /* Density check (if Q is a core point) */
                S := S ∪ N                                  /* Add new neighbors to seed set */
            }
        }
    }
}
```

Source: https://en.wikipedia.org/wiki/DBSCAN

Hyperparameters



Main hyperparameters: *min_samples* and *eps*

Other hyperparameters can be used

(using sklearn notation):

- *metric*
- *metric_params*
- *algorithm*
- *leaf_size*
- *p*
- *n_jobs*

All except n_jobs are used to specify the notion of “distance” to the model (e.g. non-euclidean)

n_jobs specify how many threads to use on the CPU (to make the algorithm possibly run faster)

DBSCAN vs K-Means

DBSCAN

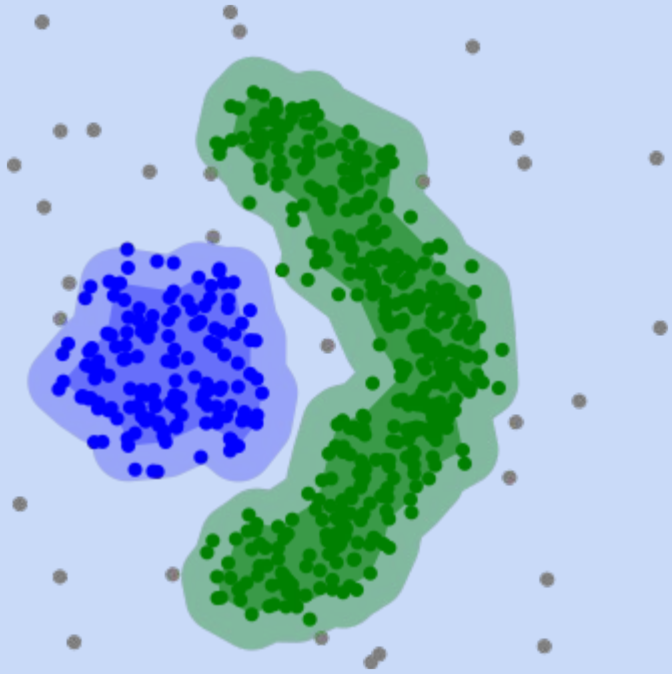
- Number of clusters determined by algorithm
- Creates clusters by finding points that are close to each other
- Separates data according to natural divisions

K-Means

- Number of clusters determined by hyperparameter *k*
- Creates clusters by finding the *centers* of potential clusters
- Separates data “linearly”

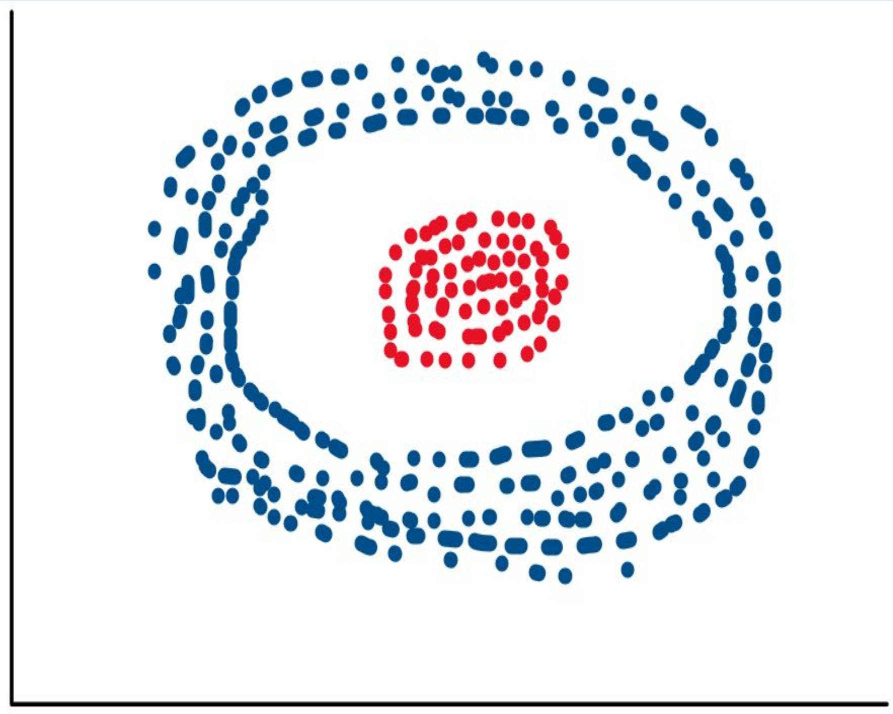
Advantages of DBSCAN

- Easily identifies noise of data
- Identifies & Excludes Outliers
- Works with uneven cluster sizes
- Deals with multidimensionality
- Data does not have to be “linearly separable”

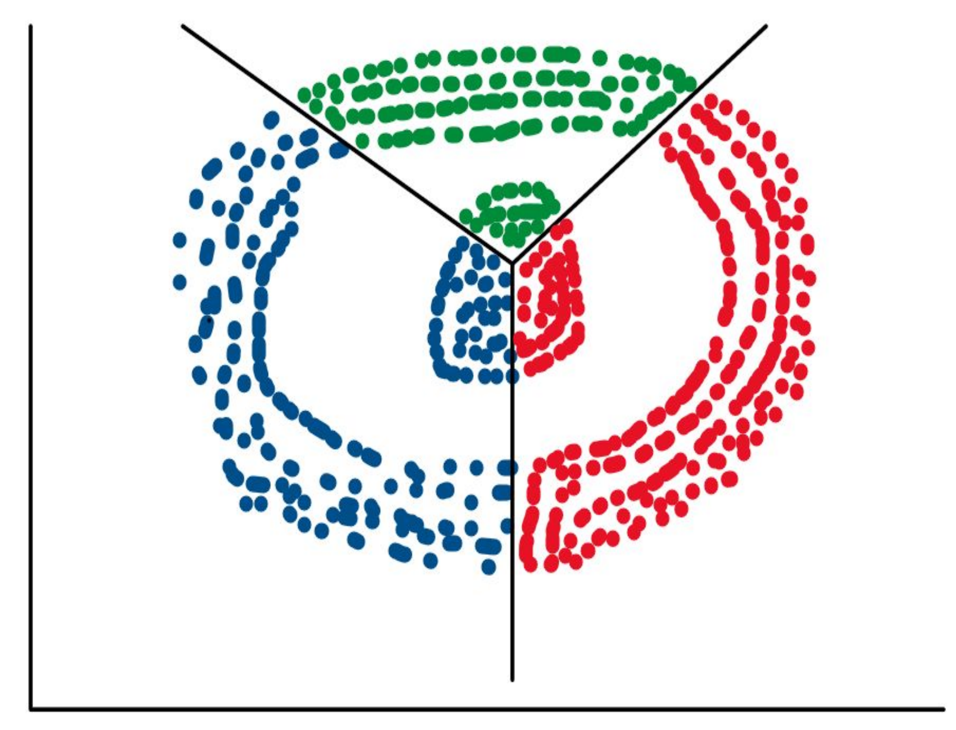


Advantages of DBSCAN

DBSCAN



K-Means



Disadvantages

- Not memory-efficient
- Less effective on high dimensional datasets due to the distance metric
- Not useful for data sets with varying densities of data
  - We can only specify 1 density (eps)!
- To detect cluster borders, the algorithm looks for some kind of density drop
  - What if the lower density data was actually part of the original cluster?

Data processing

- Standardize data to improve accuracy
  - Algorithm relies on distances
- Use as few dimensions (columns) as possible
  - If you need to use many columns, use dimensionality reduction techniques such as PCA
- No NAs please!
  - Distance cannot be calculated using null values
- Avoid replacing nulls with aggregated data
  - Because of how clustering works, replacing nulls with aggregated data will likely result in noise
- Remove duplicates (if memory is an issue)
- No need to train/test (unsupervised learning)
- No need to remove outliers since the algorithm will not group them

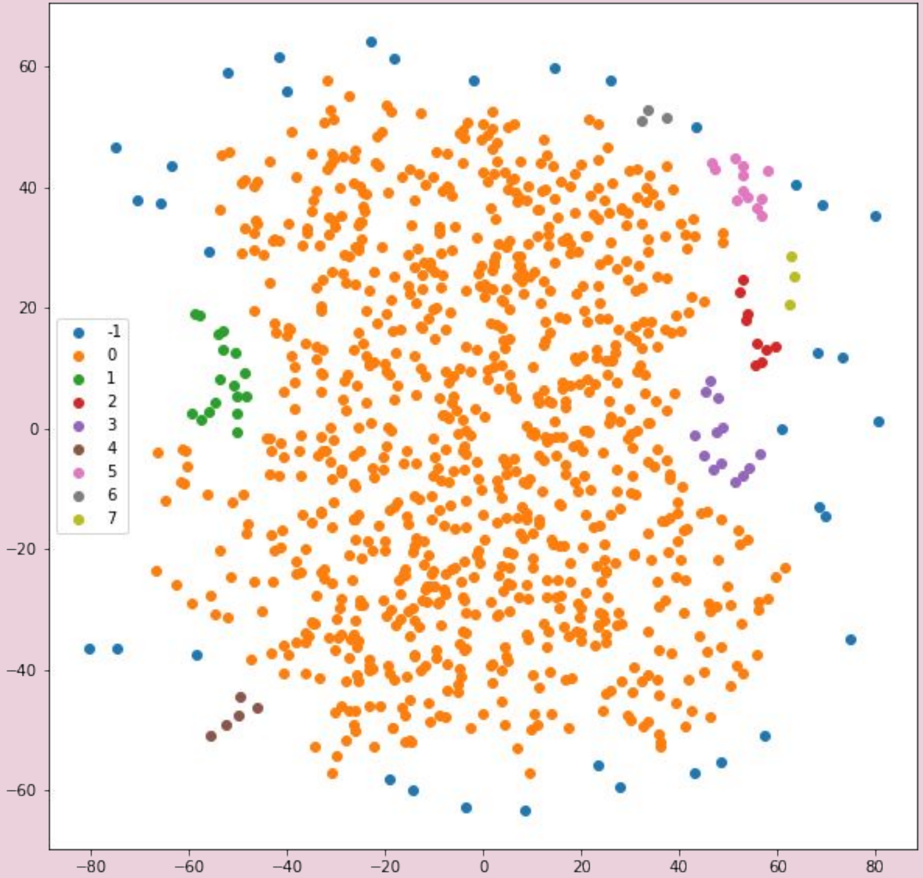
Code

```
random_data = np.random.randint(0,100,size=(1000,6))
df = pd.DataFrame(random_data, columns=['Column 1','Column 2','Column 3',
                                        'Column 4','Column 5','Column 6'])

# transforming df into 2D data
pca = PCA(2)
df = pca.fit_transform(df)

db = DBSCAN(eps=6, min_samples=3).fit(df)
label = db.labels_
u_labels = np.unique(label)
fig = plt.figure(figsize=(10, 10))
for i in u_labels:
        plt.scatter(df[label == i,0], df[label == i, 1], label = i)
plt.legend()
plt.show()
```

Output



References

https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html

https://scikit-learn.org/stable/modules/clustering.html#dbscan

sklearn.cluster.DBSCAN — scikit-learn 1.1.2 documentation

https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80

https://towardsdatascience.com/how-to-use-dbscan-effectively-ed212c02e62

https://www.youtube.com/watch?v=RDZUdRSDOok

https://www.reneshbedre.com/blog/dbscan-python.html

https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf

https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

DBSCAN Python Example: The Optimal Value For Epsilon (EPS) | by Cory Maklin | Towards Data Science

Appendix

This article gives a good overview of how DBSCAN works. It also has some code samples.
https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html

This page on scikit-learn gives a more in-depth explanation of the implementation of DBSCAN. It also touches on some of the advantages or disadvantages of the algorithm.
https://scikit-learn.org/stable/modules/clustering.html#dbscan

This is the documentation for sklearn.cluster.DBSCAN.
sklearn.cluster.DBSCAN — scikit-learn 1.1.2 documentation

This article goes over why we might want to use DBSCAN and also touches on hyperparameter estimation.
https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80

This article walks through an example implementation of DBSCAN. It also shows how to tune hyperparameters.
https://towardsdatascience.com/how-to-use-dbscan-effectively-ed212c02e62

This is a great YouTube Video from StatQuest that explains what DBSCAN algorithm does in easy to understand language and easy to follow visuals.
https://www.youtube.com/watch?v=RDZUdRSDOok

This article shows another example implementation of DBSCAN in python.
https://www.reneshbedre.com/blog/dbscan-python.html

This pdf has a nice and organized layout that explores DBSCAN on a fairly basic level and discusses good uses for DBSCAN and limitations in certain cases
https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf

This is the original paper explaining what DBSCAN is and why it works.
https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

This article describes how to use NearestNeighbors to find the optimal value for epsilon:
DBSCAN Python Example: The Optimal Value For Epsilon (EPS) | by Cory Maklin | Towards Data Science