Alper YAŞAR – 151044072

DATA MINING – HW1

DB-Scan

- The DBSCAN algorithm performs clustering by grouping areas with more objects than the threshold value, calculating the distance of objects from their neighbors and giving a specific instruction task.
- The DBSCAN algorithm is based on revealing the neighborhoods of data points with each other in two or multi-dimensional space. Database is mostly used in the analysis of spatial data because it deals with a spatial perspective.

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

DBSCAN algorithm takes Eps and MinPts values as input parameters. It checks all objects, starting with any object in the database. If the controlled object was previously included in a set, it passes to the other object without processing. If the object has not been clustered before, it finds the object's neighbors in the Eps neighborhood by doing a Region Query. If the number of neighbors is more than MinPts, this object names its neighbors as a new cluster. It then finds new neighbors by querying for a new region for each neighbor that was not previously clustered. If the neighbor numbers of the points where the region query is made more than MinPts, it is included in the cluster.

The neighborhood finding process is the part of the DBSCAN algorithm that requires the most processing power. Performance improvements to be made in this section significantly increase the performance of the algorithm. In the neighborhood study, instead of examining every point, various indexing algorithms such as R*- tree or spatial query have been proposed. With these algorithms, reduce the complexity of the DBSCAN algorithm from O (n*logn) to O (logn).

TESTS

I write algorithm in python. So I used make_blobs, make_moons for create randomly datasets.

make_blobs

Generate isotropic Gaussian blobs for clustering.

Parameters

n_samples : int, optional (default=100)

The total number of points equally divided among clusters.

n_features : int, optional (default=2)

The number of features for each sample.

centers : int or array of shape [n_centers, n_features], optional

(default=3) The number of centers to generate, or the fixed center locations.

cluster_std: float or sequence of floats, optional (default=1.0)

The standard deviation of the clusters.

random_state : int, RandomState instance or None, optional

Returns

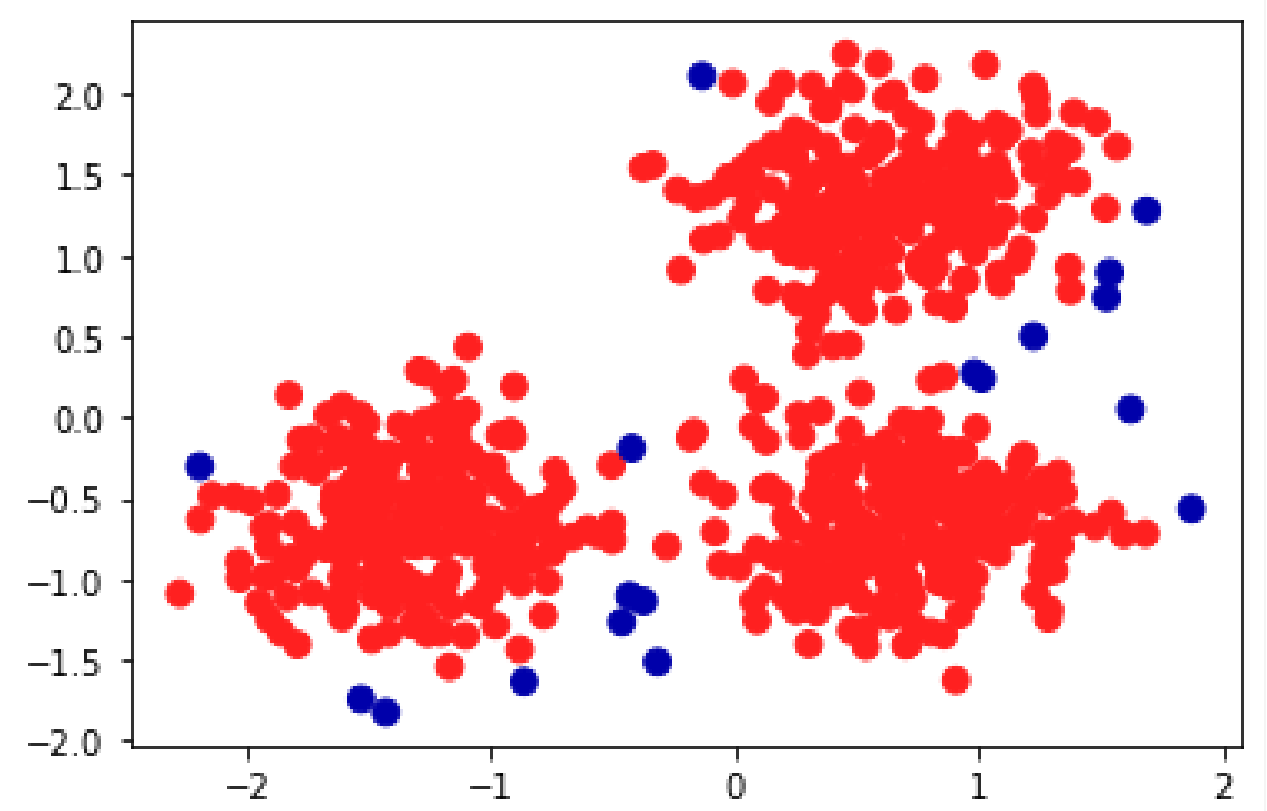
X : array of shape [n_samples, n_features]

The generated samples.

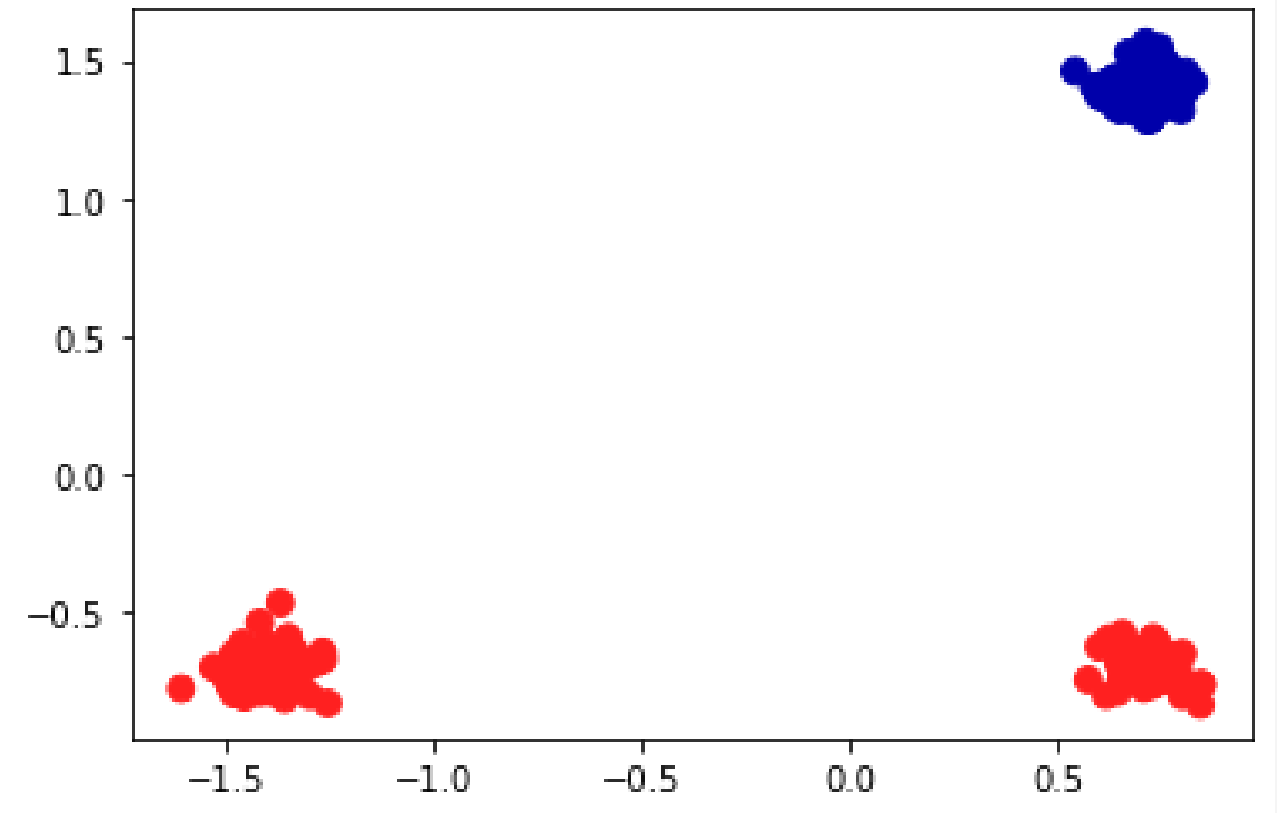
y : array of shape [n_samples]

The integer labels for cluster membership of each sample.

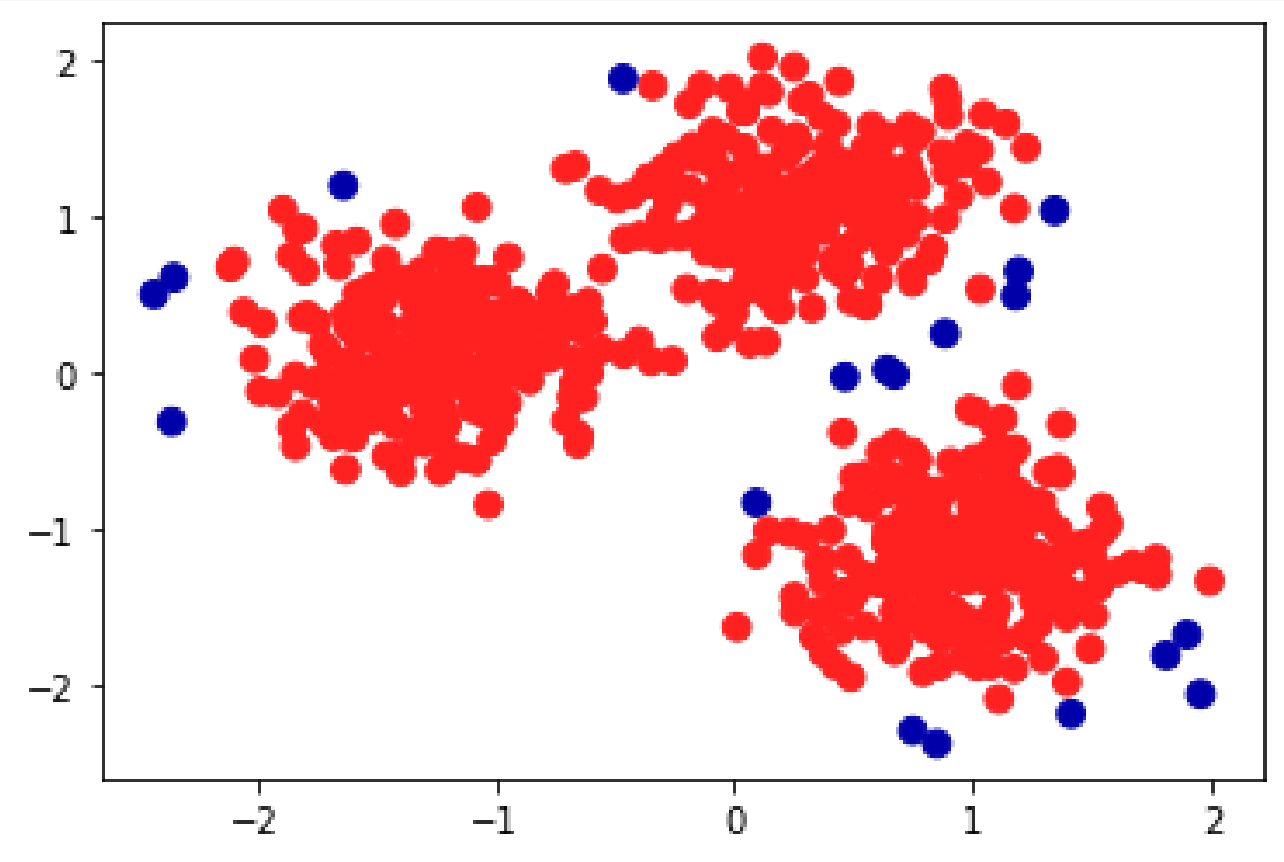
```
centers = [[1, 1], [-1, -1], [1, -1]]
make_blobs(n_samples=750, centers=centers,   cluster_std=0.4, random_state=0)
```




```
centers = [[1, 1], [-1, -1], [1, -1]]
make_blobs(n_samples=200, centers=centers, cluster_std=0.06, random_state=42)
```



make_blobs(n_samples=750, cluster_std=0.6, random_state=0)



make_moons

Make two interleaving half circles

A simple toy dataset to visualize clustering and classification algorithms.

Parameters

n_samples : int, optional (default=100)

The total number of points generated.

noise : double or None (default=None)

Standard deviation of Gaussian noise added to the data.

Returns

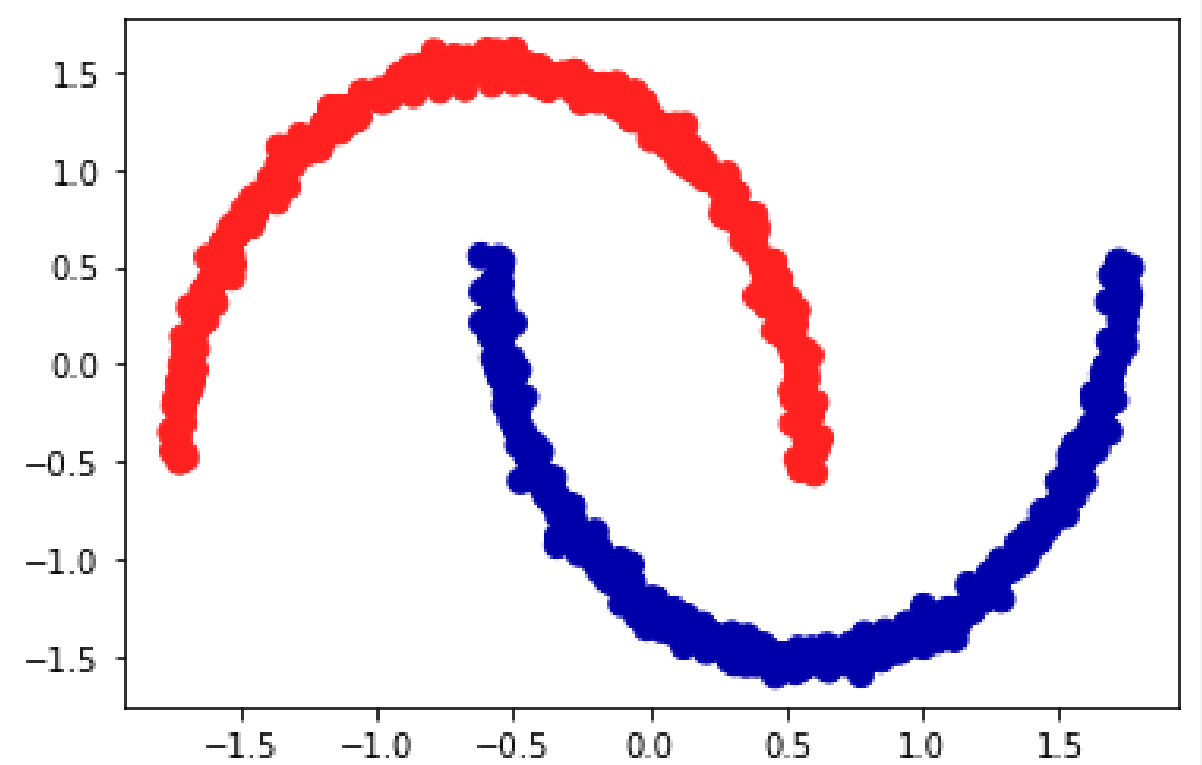
X : array of shape [n_samples, 2]

The generated samples.

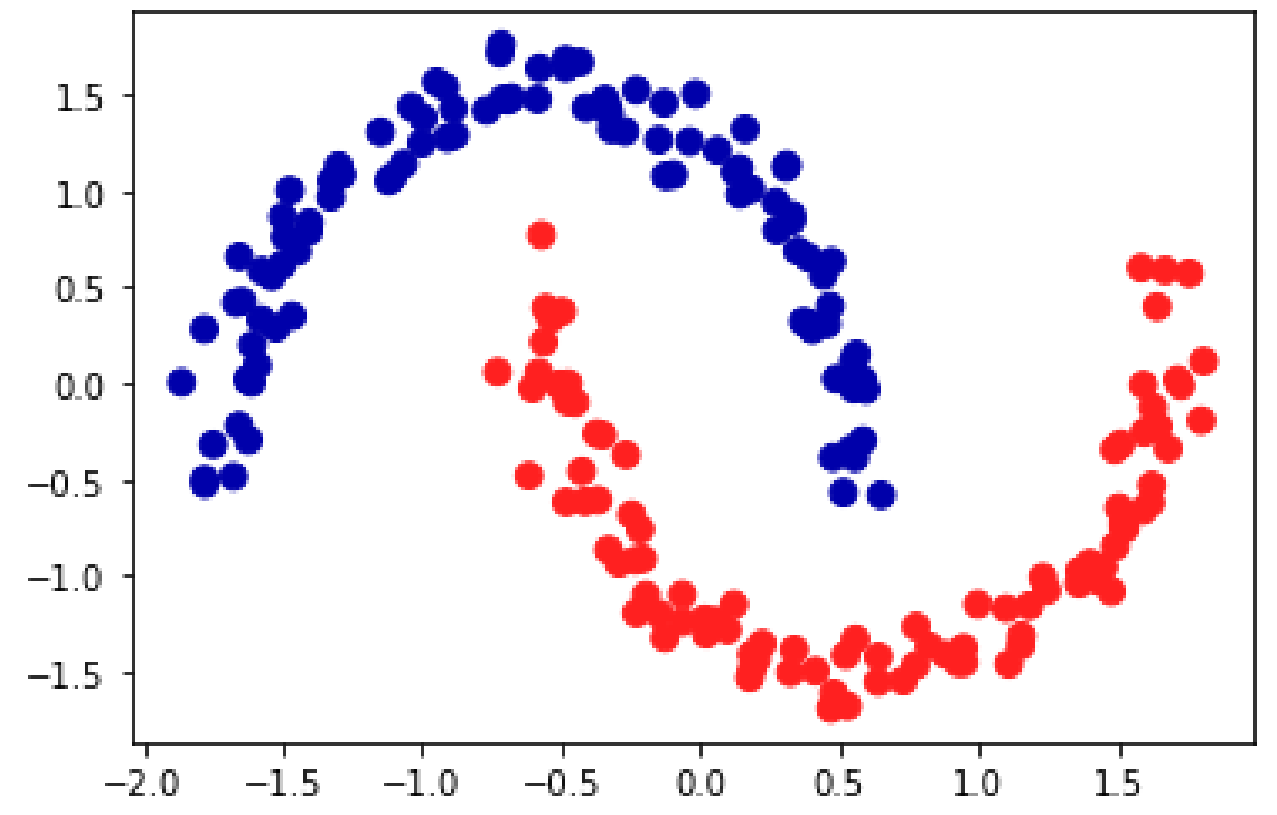
y : array of shape [n_samples]

The integer labels (0 or 1) for class membership of each sample.

make_moons(n_samples=750, noise=0.02, random_state=0)



make_moons(n_samples=200, noise=0.06, random_state=42)



RESOURCES

- Turgay Tugay BİLGİN, Yılmaz ÇAMURCU 'Applied Comparison of DBSCAN, OPTICS and K-Means Clustering Algorithms', Journal of Polytechnic Vol: 8 No: 2 pp. 139-145, 2005
- Data Mining Concepts and Techniques, Jiawei Han and Micheline Kamber, Third Edition, 2011
- https://www.youtube.com/watch?v=6BcJxBE5J5I
- https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html