DBSCAN Project Documentation

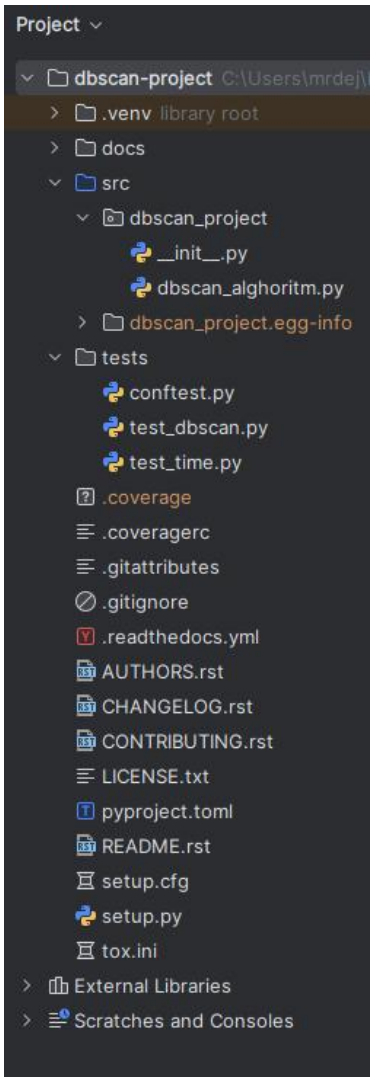
1. Overview

This project contains a simple DBSCAN (Density-Based Spatial Clustering of Applications with Noise) implementation written in Python, plus tests and a comparison with scikit-learn.

Key features

- Custom DBSCAN implementation: My_DBSCAN
- Comparison with sklearn.cluster.DBSCAN
- Tests on standard datasets: Iris, make_blobs, make_moons
- Output comparison: number of clusters, noise points, points per cluster
- Time test: custom vs sklearn

Project structure (example)



Requirements

- Python 3.x
- pytest (for running tests)
- scikit-learn (for dataset generation and comparison)

Installation (VirtualEnv example)

py -m venv .venv

Windows PowerShell:

py -m pip install -U pip

py -m pip install -e .

py -m pip install pytest scikit-learn

2. Algorithm description

DBSCAN finds clusters as connected dense regions and labels isolated points as noise.
It is useful for irregular cluster shapes and for outlier detection.

Parameters

- eps: neighborhood radius. Two points are neighbors if their distance is less than or equal to eps.
- min_samples: minimum number of neighbors required to treat a point as dense (a core point).

Point types

- Core point: has at least min_samples neighbors within eps.
- Border point: not dense enough to be core, but reachable from a core point.
- Noise point: not reachable from any cluster; labeled as -1.

High-level steps

1. Start with all points unassigned.
2. For each point, find neighbors within eps.
3. If neighbors are fewer than min_samples, mark the point as noise (-1).
4. Otherwise start a new cluster and expand it using the neighbors.

Distance metric

This project uses Euclidean distance:

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

3. User manual

Class: My_DBSCAN

File: src/dbscan_project/dbscan_alghoritm.py

Constructor

My_DBSCAN(eps=0.5, min_samples=5)
Parameters:

- eps (float > 0): neighborhood radius
- min_samples (int > 0): minimum neighbors for a core point

Input format

X should be a list of points, for example: [[x1, y1], [x2, y2], ...].
All points must have the same number of features.

Methods

- fit(X): runs clustering and stores results in self.labels and self.n_clusters; returns self.
- fit_predict(X): calls fit(X) and returns self.labels.

Output labels

- -1 means noise
- 1, 2, 3, ... are cluster IDs
- 0 is used only internally for unassigned points