Mean-shift Project documentation

This project implements the Mean Shift clustering algorithm from scratch and compares its performance and accuracy against the reference implementation in `scikit-learn`.

Requirements & Installation

This project is structured using `PyScaffold`.

Windows powershell:

virtualenv .venv

pip install numpy matplotlib scikit-learn pytest

pip install -e .

Algorithm Description Mean Shift is a non-parametric clustering algorithm that does not require prior knowledge of the number of clusters. It works by iteratively shifting data points towards the mode (the highest density of data points) in a process known as mode seeking.

Implementation Logic

1. Initialization: Every data point is treated as a starting point.

2. Shift Phase (Hill Climbing): For each point, a window of radius `bandwidth` is defined. *The "center of mass" (mean) of all neighbors within this window is calculated using a Flat Kernel. The point is shifted to this new mean. This is repeated until convergence (when points stop moving significantly).

3. Merging Phase: Points that converge to the same (or very close) locations are grouped together. These locations become the centroids.

4. Labeling: All original points are assigned to the cluster of their nearest centroid.

Usage

You can use the MyMeanShift class just like any Scikit-Learn estimator.

import numpy as np

from mean_shift_lab.mean_shift import MyMeanShift

1. Prepare data

X_train = np.array([

 [1, 1], [1.5, 1.5], [1, 2], Cluster 1

[10, 10], [10, 11], [11, 10] Cluster 2

])

2. Initialize the model

‘bandwidth' determines the radius of the search window

ms = MyMeanShift(bandwidth=2.0)

3. Fit the model to data

ms.fit(X_train)

4. Access results

print("Centroids:", ms.centroids_)

print("Labels:", ms.labels_)

Comparison & Results

We performed a comparative analysis between our custom implementation and `sklearn.cluster.MeanShift`.

Methodology

Dataset: Synthetic data generated using `make_blobs` (1000 samples, 3 centers, cluster_std=1.0).

Parameters: Both implementations used `bandwidth=1.5`.

Results

Accuracy: The custom implementation successfully identified the correct number of clusters. The calculated centroids are practically identical to the reference implementation. *

Boundaries: Decision boundaries align almost perfectly between the two models.

To reproduce the comparison experiment:

python scripts/compare_sklearn.py