Journal of Discrete Mathematics and Its Applications 10 (4) (2025) 333–358

Journal of Discrete Mathematics and Its Applications

Available Online at: http://jdma.sru.ac.ir

Shahid Rajaee Teacher Training University

Research Paper

Velvetflow: An engineering pipeline for robust multi-density clustering

Hossein Eyvazi*, Mohammad Badzohreh, Seyed Ali Shahrokhi

Department of Computer Science, University of Tarbiat Modares, Tehran, I. R. Iran

Academic Editor: Mohammad Javad Nadjafi-Arani

Abstract. Problem. Real-world datasets seldom respect a single density scale: tight blobs, elongated ribbons, and isolated points often coexist. Classical algorithms such as DBSCAN or k-means require domain-specific parameter tuning and provide only ad-hoc support for anomaly detection.

Solution. We introduce *VelvetFlow*, an *engineering pipeline* that turns a set of well-understood building blocks into a cohesive, end-to-end workflow for multi-density clustering *and* principled outlier detection. The pipeline is composed of three reusable stages: (i) *Contextual-density splitting* assigns every point to a high- or low-density partition using a single neighbourhood size k. (ii) *Density-aware clustering* applies a Jaccard-guided *FusedNeighbor*+DBSCAN routine to the sparse partition and HDBSCAN to the dense partition-without introducing new hyper-parameters. (iii) *Scaled-MST verification* re-examines the complete k-NN graph, flags weakly connected components, and validates them with a k-NN gate; this step recovers small remote clusters while filtering genuine anomalies.

Results. Across four benchmark datasets with contrasting density profiles, VelvetFlow consistently attains state-of-the-art clustering scores while producing an explicit, high-precision outlier list. (*Detailed metrics in Sec. 5.*)

Keywords. multi-density clustering, outlier detection, HDBSCAN, DBSCAN, MST, fused neighbor.
Mathematics Subject Classification (2020): 68T10, 68U20, 62H30, 62J12, 68W40.

*Corresponding author (Email address:eyvazi_hoseyn@modares.ac.ir).
Received 10 May 2025; Revised 06 July 2025; Accepted 09 August 2025
First Publish Date: 01 December 2025

1 Introduction

Clustering is a fundamental task in data science, aiming to partition datasets into groups of closely related points. Traditional clustering methods (*e.g.*, k-means, DBSCAN) often make global assumptions about data distributions, which can lead to inaccuracies when real-world datasets display heterogeneous densities, complex shapes, or significant noise. In particular, clustering in the presence of multiple densities -where some portions of the dataset are extremely dense and others are sparse or isolated- poses challenges for most single-pass algorithms.

Furthermore, outlier detection plays a decisive role in many applications, requiring robust identification of anomalous points without compromising the internal structure of valid clusters. Inconsistent densities and complex local relationships can cause misclassification of anomalies by one-size-fits-all parameters. As data grow increasingly varied, there is a need for a novel strategy that can simultaneously address multi-density clustering and reliable outlier detection.

VelvetFlow is proposed to fill this gap. Unlike single-phase strategies, VelvetFlow follows a **split-phase approach** wherein it first assigns each data point to either a high-density or low-density subset based on a carefully formulated *contextual density* metric. Points within a dense region can then be processed via hierarchical methods to capture subtle structures, while the more diffuse subset undergoes a specialized merging-based routine that unifies neighboring micro-structures without risking over-compression. Crucially, after these dedicated routines, a globally scaled *minimum spanning tree* (MST) is applied to pinpoint outlier candidates. This scaled MST offers mathematically transparent edge comparisons, making it simpler to isolate anomalies amidst varying cluster densities. A final verification uses a k-NN neighbor distance analysis to confirm or dismiss flagged candidates.

Several motivating factors drive the development of VelvetFlow:

- **Multi-density partitioning:** Traditional clustering can fail when confronted with low-density subregions adjacent to high-density cores. By separating subsets according to a local density criterion, we ensure that sparse, potentially outlier-rich areas are tackled in a manner appropriate to their inherent sparsity.

- **Adaptive local merging:** Our FusedNeighbor approach merges only sufficiently overlapping local neighborhoods. This preserves meaningful patterns in sparse regions without employing overly aggressive parameter values that might be suitable only for denser data.

- **Scalable outlier detection:** By constructing an MST on the *entire* dataset (rather than on separate regions), we exploit global geometry to extract subgraphs that significantly deviate from typical connectivity patterns. Such deviations systematically indicate outlier clusters or anomalies.

- **Minimal intrusion verification.** A lightweight verifier reuses the same neighbourhood

size k to scale MST edge lengths and applies a k-NN gate to flag outliers-*introducing no extra hyper-parameters*.

Extensive benchmarks show that *VelvetFlow*[a] outperforms both classic single-density techniques and the latest multi-density extensions in terms of clustering fidelity and outlier-detection precision. The rest of the paper is organised as follows. Section 2 surveys prior work in density-based clustering and outlier detection. Section 3 introduces the core ingredients of VelvetFlow: contextual-density scoring, the *FusedNeighbor* merge, the scaled-MST gate, and the k-NN verification step. Section 4 describes the datasets, evaluation protocol, and hyper-parameter grids, and Section 5 presents quantitative and qualitative results. Finally, Section 8 concludes with limitations and directions for future research.

2 Related work

Since 2023, density-based clustering has focused on two persistent challenges: (i) recovering clusters under strongly varying local densities and (ii) providing principled, global anomaly detection. A recent survey of Density-Peak Clustering (DPC) reviews progress up to 2023 [18]. Below we contrast *VelvetFlow* with recent DBSCAN derivatives, hierarchical multi-density models, MST hybrids, and interactive variants.

2.1 Multi-density extensions of DBSCAN

MDBSCAN and its lineage. MDBSCAN introduces a relative-density core test and a split-merge step to cope with heterogeneous densities [13]. Further work includes ADAPTIVE-DBSCAN-GASA, which couples a genetic algorithm with simulated annealing to tune $(\varepsilon, MinPts)$ [10]; AMD-DBSCAN, which retains multi-density robustness with just one user control [17]; and RDBSCAN, which employs local relative density to sharpen neighbouring cluster boundaries [20].

Despite these advances, all descendants rely on a single core/border rule and treat outliers implicitly. MDBSCAN's split-merge step also inherits parameter-sensitivity.

Our departure. *VelvetFlow* avoids any global ε by introducing a contextual-density z-score computed inside each point's k-NN shell. The same neighbourhood size k is reused for shape-aware Jaccard fusion and for a scaled-MST verification gate, so no extra hyper-parameters are added. This generalises MDBSCAN's local adaptation while improving shape preservation and offering explicit outlier control.

2.2 Hierarchical and density-peak frameworks

DMDHC prunes a linkage tree to reveal multi-density structure [5]. DPC-MFP and DPC-MDNN refine DPC with manifold or multi-feature distances, but still suffer domino-style point assignment [16, 19].

[a] Source code: `https://github.com/HosseinEyvazi/VelvetFlow`

2.3 MST-based hybrid approaches

Scaled edge weighting on a minimum spanning tree (MST) sharpens cluster cuts and flags outliers [9]. A concurrent study quantifies MST clustering's competitiveness against expert labels [11]. Our verification stage adopts these insights but reuses the same k chosen for contextual density.

2.4 Semi-supervised and active variants

Active Semi-Supervised DBSCAN (ASS-DBSCAN) injects pairwise queries to guide clustering on multi-density data [2]. Such methods assume an oracle and thus fall outside our fully unsupervised scope.

2.5 Positioning of *VelvetFlow*

VelvetFlow combines (i) contextual-density normalisation, (ii) a density-aware split-partition phase, and (iii) a scaled-MST verification gate that reuses the neighbourhood size k.

Comparison to MDBSCAN. In contrast to MDBSCAN's single relative-density threshold and post-hoc split-merge refinement, VelvetFlow

- computes a contextual-density score (a z-normalised statistic within each point's k-NN neighbourhood), eliminating $(\varepsilon, MinPts)$;
- fuses micro-partitions via a shape-aware, threshold-free Jaccard weighting on the k-NN graph; and
- verifies cluster integrity with a scaled-MST cut that reuses k, introducing no new knobs.

These steps extend MDBSCAN's local adaptation while enhancing shape robustness and providing explicit, global anomaly detection.

3 Methodology

3.1 Overview of VelvetFlow

VelvetFlow is a comprehensive clustering framework designed to effectively handle datasets with varying density distributions while simultaneously identifying outliers. The methodology is divided into distinct phases to address the challenges posed by multi-density data structures:

1. **Contextual density computation:** Assessing the local density around each data point to differentiate between high-density and low-density regions.
2. **Data partitioning:** Segregating the dataset into high-density and low-density subsets based on computed density scores.

3. **Clustering low-density subset:** Applying the *FusedNeighbor* algorithm followed by DBSCAN to identify clusters within sparse regions.

4. **Clustering high-density subset:** Utilizing HDBSCAN to uncover clusters in densely populated areas.

5. **Minimum spanning tree (MST) based outlier detection:** Constructing a scaled MST to detect anomalous points across the entire dataset.

6. **k-NN verification of outliers:** Validating outlier candidates through neighborhood distance assessments to ensure robustness.

This section elaborates on the first two phases, detailing the mathematical foundations that underpin the contextual density computation and the subsequent partitioning of data based on these metrics.

3.2 Contextual density computation

To effectively differentiate between regions of varying densities within a dataset, *VelvetFlow* employs a *contextual density* (CD) metric for each data point. This metric considers both the proximity of a point to its neighbors and the inherent density of those neighbors, providing a nuanced measure of local density.

3.2.1 Mathematical formulation

Given a dataset $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^d$, the contextual density CD_i for a point $\boldsymbol{x}_i$ is defined as

$$\mathrm{CD}_i = \frac{k}{\sum_{j=1}^{k} \max\{\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j),\ \mathrm{k\_dist}(\boldsymbol{x}_j)\}}, \tag{1}$$

where

- d is the dimensionality of the feature space (number of attributes per point);
- k is the number of nearest neighbours considered;
- $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$;
- $\mathrm{k\_dist}(\boldsymbol{x}_j)$ is the distance from $\boldsymbol{x}_j$ to its k-th nearest neighbour in $\mathcal{X}$.

Interpretation: A higher CD_i value signifies that $\boldsymbol{x}_i$ is located in a region with dense clustering, as it is surrounded by neighbors that are both close in proximity and reside in inherently dense areas themselves. Conversely, a lower CD_i indicates that $\boldsymbol{x}_i$ is in a sparser region, possibly isolated or part of a less dense cluster.

3.2.2 Normalization

To ensure that the contextual density scores are comparable across different datasets and scales, normalization is applied:

$$\mathrm{CD}_i^{\mathrm{norm}} = \frac{\mathrm{CD}_i - \min(\mathrm{CD})}{\max(\mathrm{CD}) - \min(\mathrm{CD})}, \tag{2}$$

where $\min(\mathrm{CD})$ and $\max(\mathrm{CD})$ are the minimum and maximum contextual density values across all points in $\mathcal{X}$, respectively.

3.3 Data partitioning

Utilizing the computed contextual density scores, the dataset is bifurcated into high-density and low-density subsets. This partitioning facilitates tailored clustering strategies appropriate for each density regime.

3.3.1 Threshold determination

A threshold parameter $t \in (0,1)$ is selected to delineate high-density regions from low-density ones. Points with normalized contextual density exceeding this threshold are classified as high-density, while the rest are deemed low-density:

$$\boldsymbol{x}_i \in \mathcal{X}_H \quad \text{if} \quad \mathrm{CD}_i^{\mathrm{norm}} > t, \tag{3}$$

$$\boldsymbol{x}_i \in \mathcal{X}_L \quad \text{if} \quad \mathrm{CD}_i^{\mathrm{norm}} \leq t. \tag{4}$$

3.3.2 Mathematical justification

The selection of the threshold t is pivotal for effective partitioning. A mathematically sound approach involves analyzing the distribution of $\mathrm{CD}_i^{\mathrm{norm}}$ across $\mathcal{X}$. Techniques such as the *elbow method* or *knee detection algorithms* can be employed to identify an optimal t where the rate of change in density scores significantly alters, indicating a natural separation between dense and sparse regions.

3.3.3 Partitioning algorithm

The partitioning process can be formalized as follows:

3.4 Clustering low-density subset

The low-density subset $\mathcal{X}_L$ often contains sparse clusters or isolated points that require a specialized clustering approach to accurately identify meaningful groupings without being overwhelmed by noise. *VelvetFlow* employs the *FusedNeighbor* algorithm in conjunction with DBSCAN to address this challenge.

3.4.1 FusedNeighbor algorithm

FusedNeighbor is designed to merge points in $\mathcal{X}_L$ based on the similarity of their local neighborhood structures. The algorithm operates as follows:

Algorithm 1 Data partitioning based on contextual density

Require: Dataset $\mathcal{X}$, contextual density scores $\{\mathrm{CD}_i^{\mathrm{norm}}\}$, threshold t
Ensure: High-density subset $\mathcal{X}_H$, Low-density subset $\mathcal{X}_L$

```
X_H ← ∅
X_L ← ∅
for each point x_i ∈ X do
    if CD_i^norm > t then
        X_H ← X_H ∪ {x_i}
    else
        X_L ← X_L ∪ {x_i}
    end if
end for
return (X_H, X_L)
```

1. **k-Nearest Neighbors identification:** For each point $\boldsymbol{x}_i \in \mathcal{X}_L$, identify its k-nearest neighbors within $\mathcal{X}_L$:
$$\mathrm{kNN}(\boldsymbol{x}_i) = \{\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \ldots, \boldsymbol{x}_{j_k}\}.$$

2. **Jaccard similarity computation:** Compute the Jaccard similarity between the neighbor sets of each pair of points:
$$\mathrm{Jaccard}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|\mathrm{kNN}(\boldsymbol{x}_i) \cap \mathrm{kNN}(\boldsymbol{x}_j)|}{|\mathrm{kNN}(\boldsymbol{x}_i) \cup \mathrm{kNN}(\boldsymbol{x}_j)|}.$$

3. **Merging criterion:** If $\mathrm{Jaccard}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \tau$, where τ is a predefined similarity threshold, merge the clusters containing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

4. **Iterative merging:** Repeat the merging process iteratively until no further merges satisfy the Jaccard similarity criterion.

3.4.2 Mathematical justification

The rationale behind using Jaccard similarity lies in its ability to quantify the overlap between the neighborhoods of two points, effectively capturing the structural similarity within $\mathcal{X}_L$. By setting a threshold τ, we ensure that only points with sufficiently similar local environments are merged, thereby preserving the integrity of distinct sparse clusters and minimizing the amalgamation of unrelated points.

3.4.3 DBSCAN refinement

After the *FusedNeighbor* merging process, the resultant clusters in $\mathcal{X}_L$ may still contain minor variations or outliers. To refine these clusters, DBSCAN is applied with parameters tailored for low-density regions:

- $\varepsilon_{\mathrm{fused}}$: A smaller neighborhood radius suitable for sparse data.

- $\text{minPts}_{\text{fused}}$: A minimal number of points to form a core point in the context of low-density clusters.

The application of DBSCAN post-*FusedNeighbor* ensures that any residual outliers are effectively identified and that clusters within $\mathcal{X}_L$ are cohesively formed without being dominated by noise.

3.4.4 Algorithmic representation

Algorithm 3 summarises the full method. It computes contextual density, splits data into homogeneous regions, merges them based on shared k-NNs, and verifies outliers using a scaled MST. All steps use the same k and no extra hyper-parameters are introduced. The total complexity is $O(N \log N + Nk)$.

Algorithm 2 FusedNeighbor clustering on low-density subset

Require: Low-density subset $\mathcal{X}_L$, number of neighbors k, Jaccard threshold τ
Ensure: Merged cluster indices C_M, Remaining point indices C_R

// Each point is initially in its own cluster
for all $\boldsymbol{x}_i \in \mathcal{X}_L$ **do**
    assign a new singleton cluster to $\boldsymbol{x}_i$
end for
// Jaccard-based merging over all pairs
for all pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with $i < j$ **do**
    $\text{Jaccard}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leftarrow \frac{|\text{kNN}(\boldsymbol{x}_i) \cap \text{kNN}(\boldsymbol{x}_j)|}{|\text{kNN}(\boldsymbol{x}_i) \cup \text{kNN}(\boldsymbol{x}_j)|}$
    if $\text{Jaccard}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \tau$ **then**
        merge the clusters containing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
    end if
end for
// Remove empty clusters after merges
prune all empty clusters from the cluster list
// Compute mean cluster size
$\mu \leftarrow \frac{1}{|C_M|} \sum_{c \in C_M} |c|$
// Identify large clusters
$C_{\text{large}} \leftarrow \{ c \in C_M \mid |c| \geq \mu \}$
$C_M \leftarrow C_{\text{large}}$
$C_R \leftarrow \mathcal{X}_L \setminus \bigcup C_M$
return (C_M, C_R)

3.4.5 Parameter selection

The effectiveness of *FusedNeighbor* is contingent upon the appropriate selection of the number of neighbors k and the Jaccard similarity threshold τ. Empirical tuning based on cross-validation or domain-specific insights is recommended to optimize cluster formation and outlier exclusion.

Hyperparameters

- k neighbourhood size used (i) to compute contextual density and (ii) again in the k-NN outlier-verification step.
- threshold on the z-normalised density that divides the data into low-density (X_L) and high-density (X_H) subsets.
- ε_L, $\mathbf{MinPts}_L$ DBSCAN parameters applied only to the low-density subset after the FusedNeighbor pre-step.
- $\mathbf{min\_cluster\_size}_H$, $\mathbf{min\_samples}_H$ HDBSCAN parameters used only for clustering the high-density subset.
- **MST-based outlier detection** - uses the scaled MST; no additional parameters.
- k **reused** the same k from the first item is reused for verifying candidate outliers, so no new knob is introduced at this stage.

Early choices (k, t) influence which points flow into later stages, while the DBSCAN and HDBSCAN settings control cluster detail within their respective density regimes. This list shows exactly where each parameter enters the pipeline.

3.4.6 Mathematical guarantees

By leveraging Jaccard similarity for cluster merging, *FusedNeighbor* ensures that only points with a substantial overlap in their local neighborhoods are combined. This approach preserves the structural integrity of distinct clusters within $\mathcal{X}_L$ and mitigates the risk of merging unrelated or sparsely connected points, thereby enhancing the overall robustness of the clustering process.

Note: The subsequent part of the methodology will detail the clustering of the high-density subset $\mathcal{X}_H$, the construction and scaling of the Minimum Spanning Tree (MST) for outlier detection, and the final consolidation of cluster labels.

3.5 Clustering high-density subset

Once the low-density subset $\mathcal{X}_L$ has been processed, we focus on the high-density subset $\mathcal{X}_H$, whose points $\boldsymbol{x}_i$ satisfy $\mathrm{CD}_i^{\mathrm{norm}} > t$. In this regime, a single global parameter is typically less problematic because the points are already densely packed. However, a fixed ε might still merge distinct dense clusters or exclude fine boundary variations. To address this, *VelvetFlow* adopts a hierarchical approach, leveraging **HDBSCAN**:

1. **Core distances and mutual reachability:** HDBSCAN first computes a *core distance* for each point $\boldsymbol{x}_i \in \mathcal{X}_H$, given by

$$\mathrm{core\_dist}(\boldsymbol{x}_i) \;=\; \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_{(k)})$$

where $\boldsymbol{x}_{(k)}$ is the k-th nearest neighbor in the dense region (or *MinPts* if k is not used). Mutual reachability between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus becomes:

$$\text{mreach}(\boldsymbol{x}_i, \boldsymbol{x}_j) \;=\; \max\Big(\text{core\_dist}(\boldsymbol{x}_i), \text{core\_dist}(\boldsymbol{x}_j), \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big).$$

2. **Hierarchy construction:** By constructing a *minimum spanning tree* of mutual reachability distances, HDBSCAN generates a hierarchy (condensed tree) that gradually reveals cluster splits at varying densities. Mathematically, smaller edges highlight robust connections within dense regions, while larger edges correspond to sparser bridges.

3. **Cluster extraction:** A stability measure is used to decide which branches of the condensed tree represent stable clusters at different density levels. Points that do not fit well into any stable branch become noise in the high-density context.

Advantages for high-density data: This hierarchical approach sidesteps the need for a singular ε, thereby mitigating the risk of under- or over-clustering in $\mathcal{X}_H$. It automatically adapts to varied microstructures within dense regions, capturing subtle local densities that a global parameter might obscure.

3.6 Global MST for outlier detection in multi-density context

After separately clustering $\mathcal{X}_L$ and $\mathcal{X}_H$, we obtain two sets of cluster labels and merge them into a unified label set (using unique integers). However, to identify outliers *across all densities*, it is essential to capture abrupt transitions between clusters of varying density. Hence, *VelvetFlow* constructs a *scaled Minimum Spanning Tree (MST)* over *all* points in $\mathcal{X}$-both low- and high-density subsets. This global MST highlights sudden increases in distance that likely occur at the interfaces of multi-density clusters or truly isolated anomalies.

3.6.1 Motivation for scaled MST

In multi-density scenarios, each dense cluster can exhibit its own characteristic distance scale. Constructing a Minimum Spanning Tree (MST) over the entire dataset, therefore, provides a single global structure that encapsulates the connectivity across varying densities. By *scaling* each edge in the MST, we emphasize *relative* rather than absolute distance jumps. This scaling ensures that transitions from one density cluster to another manifest as unusually large scaled edges. Consequently, this multi-density-aware approach effectively flags points that do not integrate smoothly into any local cluster core, thereby identifying potential outliers irrespective of their local density.

3.6.2 MST construction and edge scaling

Let $\mathcal{T}$ be the MST constructed over all N points in $\mathcal{X}$. To construct $\mathcal{T}$, *VelvetFlow* employs Prim's algorithm, which begins with the smallest edge and progressively adds the next smallest edge that connects a new node to the growing MST. Suppose the edges selected in the MST are ordered as $\{e_1, e_2, \ldots, e_{N-1}\}$ with ascending weights $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_{N-1})$. We define the *scaled MST* $\mathcal{T}_s$ by assigning each edge e_i a scaled weight $w_s(e_i)$ as follows:

$$w_s(e_i) = \begin{cases} w(e_1), & \text{if } i = 1, \\ \dfrac{w(e_i)}{w(e_{i-1})}, & \text{if } i > 1. \end{cases} \tag{5}$$

For $i > 1$, the scaled weight $w_s(e_i)$ compares the current edge's weight $w(e_i)$ to the previous edge's weight $w(e_{i-1})$. Edges that represent unusually large jumps-indicative of boundaries between dense regions or connections to outliers-will thus have significantly higher scaled weights $w_s(e_i)$. This scaling process accentuates transitions in the MST that are characteristic of multi-density structures, facilitating the identification of outlier candidates.

3.6.3 Cutting the MST and identifying candidates

To detect potential outliers, we process the scaled MST $\mathcal{T}_s$ by performing the following steps:

1. **Sorting scaled edges:** Sort all edges of $\mathcal{T}_s$ in descending order based on their scaled weights $w_s(e_i)$.

2. **Iterative cutting:** Iterate through the sorted edges and perform *iterative cuts* by removing the edge with the highest scaled weight:

   $$\text{Cut}(\mathcal{T}_s,\, e_i) \;\longrightarrow\; (\mathcal{T}_1,\, \mathcal{T}_2),$$

   where $\mathcal{T}_1$ and $\mathcal{T}_2$ are the resulting connected components after removing edge e_i.

3. **Applying stopping criterion:** Define a size threshold minnormal, representing the minimum number of points required for a subgraph to be considered "normal." If removing an edge e_i results in a subgraph S such that $|S| <$ minnormal, mark the entire subgraph S as potentially outlier-dense. This is based on the rationale that extremely small connected components typically form at regions with sharp density changes, indicating isolated or boundary-like points.

In multi-density datasets, these small components often emerge at steep transitions between clusters of differing densities, thereby signaling anomalous or boundary points that warrant further examination as outliers [8].

3.6.4 k-NN verification of outliers

The *scaled* MST exaggerates relative edge lengths so that unusually long links stand out, enabling a single global cut to expose weakly connected objects. A side-effect, however, is that a *legitimate* cluster that is both very small and far from the rest of the data may attach through one overscaled edge (Fig. 1). Removing that edge places every point in the micro-cluster into the outlier candidate set $\mathcal{O}$.

To prevent such false positives, VelvetFlow performs an additional k-Nearest-Neighbor (k-NN) validation:

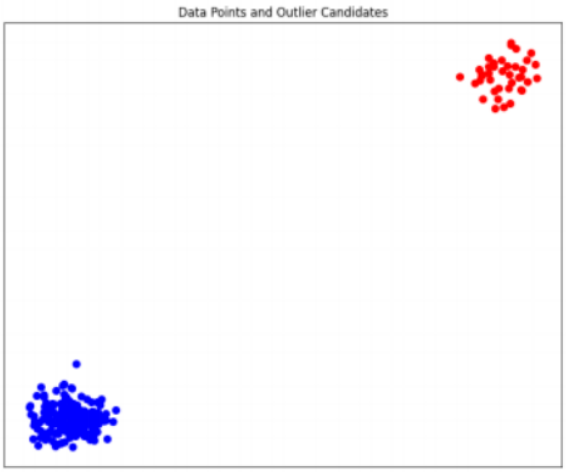

Figure 1. Illustrates a specific problem encountered when using the scaled minimum spanning tree (MST) approach, particularly in outlier detection. The core issue highlighted is the incorrect identification of a smaller, distant cluster as outliers.

1. **Neighbourhood reuse.** The same neighbourhood size k selected for contextual density and *FusedNeighbor* is reused, ensuring a uniform locality scale.

2. **Cluster map.** Let $C = \{C_1, C_2, \dots\}$ denote the current clusters after merging the low- and high-density partitions.

3. **Average local distance.** For each cluster C_i, compute

$$\overline{D}_{C_i} = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \text{avgKnnDist}(\boldsymbol{x}),$$

   the mean k-NN distance within that cluster.

4. **Candidate check.** For every $\boldsymbol{x} \in \mathcal{O}$ with provisional label C_i, test

$$\text{avgKnnDist}(\boldsymbol{x}) > \alpha \, \overline{D}_{C_i}.$$

   If true, $\boldsymbol{x}$ is confirmed as an outlier and assigned label -1; otherwise it is re-integrated into C_i.

Because points in a remote but internally coherent micro-cluster have neighbour distances similar to $\overline{D}_{C_i}$, they pass the test and are retained as a bona-fide cluster. Truly isolated points, whose neighbour distances far exceed the cluster mean, remain labelled as anomalies. In this manner, the global sensitivity of the scaled MST is balanced by a local, density-aware verification step, yielding robust outlier detection across heterogeneous densities.

3.7 Label consolidation and final output

After verifying outliers, *VelvetFlow* combines the clusters from:

- **Low-density subset clusters:** Identified through FusedNeighbor + DBSCAN.
- **High-density subset clusters:** Discovered by HDBSCAN.
- **Verified outliers:** Marked with label -1.

Label Unification. We assign unique integer labels to each cluster, ensuring no overlap between the sets originating from $\mathcal{X}_L$ and $\mathcal{X}_H$. All verified outliers remain labeled -1. This final set of labels, denoted $\boldsymbol{y}_{\text{Velvet}}$, is the ultimate output of the method.

3.8 Overall algorithmic workflow

The VelvetFlow pipeline, illustrated in Figure 2, processes multi-density datasets through sequential stages including density classification, fused neighbor detection, and cluster merging.

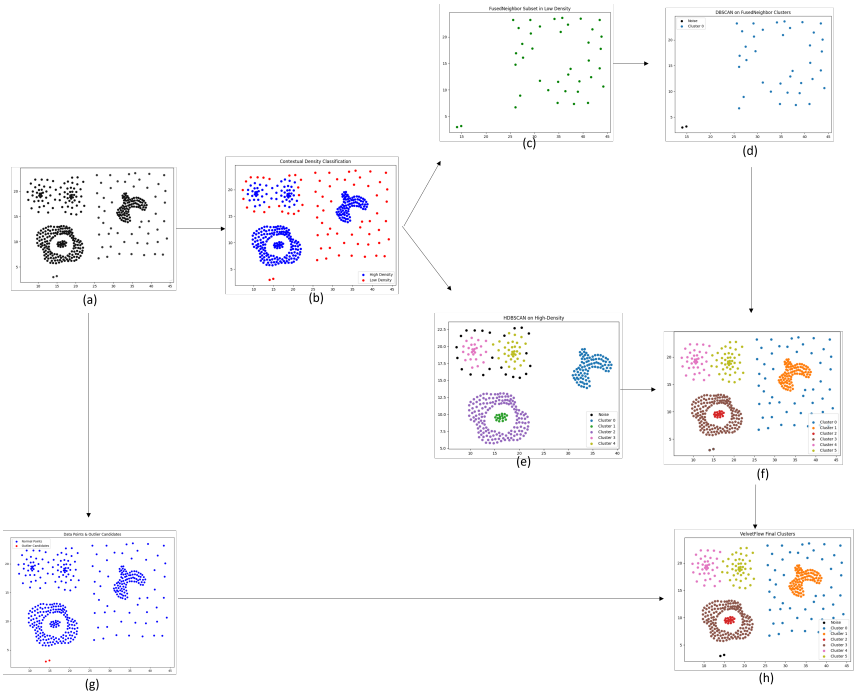

Figure 2. **VelvetFlow pipeline overview.** The clustering process of *VelvetFlow* is depicted through sequential steps labeled (a) to (h). This pipeline effectively handles multi-density clusters by segregating high and low-density regions, employing specialized clustering techniques for each subset, and utilizing MST-based methods for robust outlier detection and cluster consolidation.

VelvetFlow pipeline steps:

1. **Compound dataset:** The initial dataset containing clusters with varying densities.

2. **Density classification:** Classification of data points into high-density ($\mathcal{X}_H$) and low-density ($\mathcal{X}_L$) subsets based on contextual density scores.
3. **FusedNeighbor identification:** Detection of fused neighbors within the low-density subset to identify potential micro-clusters.
4. **DBSCAN on fused neighbors:** Application of DBSCAN clustering on the merged neighbors obtained from step (c).
5. **HDBSCAN on high-density subset:** Execution of HDBSCAN clustering on the high-density subset classified in step (b).
6. **Merging clusters:** Integration of clusters from steps (d) and (e), assigning noise points to the nearest clusters and incorporating non-fused points from the low-density subset.
7. **Scaled MST outlier detection:** Construction of a scaled Minimum Spanning Tree (MST) on the entire dataset to identify outlier candidates through edge scaling and iterative cutting.
8. **Final cluster formation:** Verification and confirmation of outliers using k-NN within their respective clusters, resulting in the final set of consolidated clusters.

Algorithm 3 succinctly captures these steps.

Algorithm 3 VelvetFlow clustering algorithm

Require: Dataset $\mathcal{X}$, parameters $(k, t, \varepsilon_{\text{fused}}, \text{minPts}_{\text{fused}}, \dots)$
Ensure: Final labels $\boldsymbol{y}_{\text{Velvet}}$ of size $|\mathcal{X}|$

$\text{CD}_i \leftarrow \text{ComputeContextualDensity}(\mathcal{X}, k)$
$(\mathcal{X}_H, \mathcal{X}_L) \leftarrow \text{PartitionByDensity}(\mathcal{X}, \{\text{CD}_i\}, t)$
$(\text{mergedLow}, \text{remainLow}) \leftarrow \text{FusedNeighbor}(\mathcal{X}_L, k)$
$\text{DBSCANLabels}_L \leftarrow \text{RefineLowDensityClusters}(\text{mergedLow}, \varepsilon_{\text{fused}}, \text{minPts}_{\text{fused}})$
$\text{HDBSCANLabels}_H \leftarrow \text{ClusterHighDensity}(\mathcal{X}_H)$
$\text{rawLabels} \leftarrow \text{MergeLabels}(\text{DBSCANLabels}_L, \text{HDBSCANLabels}_H)$
$\text{rawLabels}[\text{remainLow}] \leftarrow -1$ ▷ Mark leftover low-density as noise
$(\text{MST}, \text{edges}) \leftarrow \text{constructMST}(\mathcal{X})$
$\text{MST}_{\text{scaled}} \leftarrow \text{scaledMST}(\text{MST}, \text{edges})$
$\mathcal{O} \leftarrow \text{ExtractCandidates}(\text{MST}_{\text{scaled}})$
$\text{verifiedOutliers} \leftarrow \text{verifyOutliers}(\text{rawLabels}, \mathcal{O})$
Mark verifiedOutliers **as** -1
return rawLabels

Mathematically, each component (local density partitioning, FusedNeighbor + DBSCAN in sparse zones, HDBSCAN in dense zones, and MST-based outlier detection) ensures a multi-stage approach to capturing clusters that vary in scale, while systematically isolating anomalies. This integrated pipeline has demonstrated strong performance across datasets with distinct density profiles, as will be shown in the subsequent experimental results.

3.9 Time complexity analysis

To complete our methodological discussion, we summarize the computational complexity of *VelvetFlow* in its main stages. The actual runtime may vary depending on data structures (e.g., kd-trees, approximate neighbor search), dimensionality d, and the fraction of points classified as low- or high-density. We denote N as the total number of points in $\mathcal{X} \subset \mathbb{R}^d$.

1) Computing contextual density.

- **Naive:** Pairwise distance checks for all N points incur $O(N^2)$.
- **Optimized:** Using spatial indices (kd-tree, ball tree) can reduce neighbor searches to $O(N \log N)$ for low to moderate d. Computing the local reachability metrics and normalization then adds $O(N \cdot k) + O(N)$, which is dominated by $O(N \log N)$ under a good search structure.

2) Partitioning by density. Comparing each point's density score to a threshold t is an $O(N)$ operation.

3) FusedNeighbor on low-density subset. Let M be the number of points in the low-density subset (so $M \leq N$).

- **Neighbor construction:** Building or re-indexing M points can cost $O(M^2)$ naively or $O(M \log M)$ with optimized structures.
- **Jaccard-based merging:** In the worst case, checking pairwise Jaccard between M points is $O(M^2 \times k)$. Converting to a graph approach, we might store up to $O(M^2)$ edges, making the connected component merges $O(M^2)$.
- **Overall:** If M is large (close to N), the step can hit $O(N^2)$ in the worst case. However, if $M \ll N$, it is significantly faster.

4) DBSCAN Refinement on fused clusters. Applying DBSCAN on each fused cluster of size $M' \leq M$ yields $O(M'^2)$ naively or $O(M' \log M')$ with indexing. Summed across multiple smaller clusters, the total remains $\leq O(M^2)$ (naive) or closer to $O(M \log M)$ if clusters are well separated.

5) HDBSCAN on high-density subset. Let the high-density portion have $(N - M)$ points:

- **Naive:** Building a mutual reachability graph and hierarchical tree can reach $O((N - M)^2)$ if we rely on all pairwise distances.
- **Optimized:** In low d, specialized structures can reduce it to $O((N - M) \log(N - M))$, but practically $O(N^2)$ remains an upper bound for general dimensions.

6) Scaled MST for outlier detection. Constructing an MST on the full dataset:

- **Naive:** $O(N^2)$ for building the complete graph, plus $O(N^2)$ for standard MST algorithms (like Prim's or Kruskal's on N^2 edges).
- **Optimized (Using Prim's Algorithm):** Since *VelvetFlow* employs Prim's algorithm for MST construction, the time complexity can be improved with appropriate data structures:
  - **Using a binary heap:** Prim's algorithm runs in $O(N^2)$ time when implemented with a simple binary heap, which is suitable for dense graphs.
  - **Using a Fibonacci heap:** For sparse graphs, Prim's algorithm with a Fibonacci heap can achieve $O(E + N \log N)$ time complexity, where E is the number of edges. However, in the context of constructing a complete graph, $E = O(N^2)$, and the complexity remains $O(N^2)$.
- **Edge scaling & cutting:** Sorting $N - 1$ edges is $O(N \log N)$, and iterative cuts plus subgraph checks is $O(N)$ or $O(N \log N)$ overall.

7) k-NN verification of outliers. For each candidate outlier, verifying with a local k-NN distance check inside its assigned cluster incurs $O(k)$ per point once indexing is available, or $O(\log N)$ if a tree-based search is used. Overall, this step is typically subsumed by the preceding MST construction's complexity.

Summary of overall complexity.

- **Naive bound:** $O(N^2)$, typically dominated by MST construction or pairwise neighbor searches.
- **Optimized bound:** $O(N \log N)$ for lower-dimensional data using advanced techniques (e.g., kd-trees, Delaunay-based MST), though $O(N^2)$ can reappear if d is high or if many subset merges demand dense graph computations.

In practice, we often observe performance closer to $O(N \log N)$ for moderate N (up to a few tens of thousands) and low d, especially if approximate neighbor searches or parallelization are employed. Nevertheless, $O(N^2)$ remains a safe upper bound for *VelvetFlow*, particularly if the dimension is large or $M \approx N$ for the low-density partition.

4 Experimental setup

4.1 Datasets and ground truth

For our evaluation, we select four representative datasets that exhibit varying densities, complex boundary shapes, and potential outliers:

1. **Pathbased:** Known for narrow curvilinear structures with multiple density levels.
2. **Compound:** Contains several clusters of diverse densities, including an elongated region and a few compact groups.
3. **Jain:** Emphasizes strong density contrasts between two clusters.
4. **Synthetic (Multi-density):** An internally generated dataset combining multiple density clusters and additionally featuring a *very sparse cluster* scattered across the entire plane. This setup is intended to highlight our algorithm's robustness in handling both dense regions and exceedingly sparse outliers.

Each dataset is assumed to contain ground truth labels in its third column (the first two columns represent the coordinates). A typical example of formatting is:

Feature1	**Feature2**	**GroundTruth**
$x_{1,1}$	$x_{1,2}$	g_1
$x_{2,1}$	$x_{2,2}$	g_2
$\vdots$	$\vdots$	$\vdots$

We evaluate clustering performance by comparing predicted labels with these ground truth labels.

4.2 Implementation details

All algorithms, including our proposed *VelvetFlow* method, are implemented in `Python`. We leverage libraries such as `scikit-learn` (for `DBSCAN` and `OPTICS`), and `hdbscan` (for `HDBSCAN`). The `networkx` library handles MST construction. Our experiments ran on a standard workstation equipped with 16 GB RAM and an 8-core Intel Core i7-9700K CPU (3.6 GHz base clock). For data containing N points in $\mathbb{R}^2$, *VelvetFlow* typically operates in $O(N \log N)$ or $O(N^2)$ depending on MST construction specifics (exact or approximate).

4.3 Parameter settings and baselines

We compare *VelvetFlow* against the following density-based baselines:

- **DBSCAN** [7]: tuned over a modest logarithmic grid for the radius parameter and across a small range of core-point counts.
- **HDBSCAN** [12]: varied by several choices of *min-cluster-size* and *min-samples*, from very small to moderately large values.
- **MDBSCAN** [13]: first sweeps a handful of neighbourhood sizes to stabilise the relative-density histogram, selects the split threshold at the first clear valley of that histogram, and finally refines the internal DBSCAN parameters on the same qualitative grid used for vanilla DBSCAN.

Hyper-parameter search. All baselines receive an equal-budget grid or local search:

- *DBSCAN*: explores a radius grid centred on the k-dist elbow, with core-point counts spanning low to medium values.
- *HDBSCAN*: evaluates several small, medium and relatively large settings for each of its two key parameters.
- *MDBSCAN*: uses the neighbourhood sweep described above, sets the density split where the empirical histogram shows a marked drop, then refines its internal DBSCAN parameters on the same qualitative grid as standard DBSCAN.

VelvetFlow requires tuning only a single neighbourhood size k and a density split threshold t; all later stages reuse these values, eliminating the need for further parameter search.

5 Results

This section quantifies the performance of *VelvetFlow* against three competitive density-based baselines-**DBSCAN**, **MDBSCAN**, and **HDBSCAN**-on four public benchmarks that span a wide range of cluster shapes and density contrasts. All runs share the same hardware platform described in Sec. 4.2; baseline grids are reported in the supplementary material. To keep the presentation focused, we highlight three metrics that jointly capture partition integrity (**NMI**), pairwise agreement (**ARI**), and fine-grained boundary accuracy (**Pairwise** F_1).[b]

5.1 Evaluation metrics

- **Normalised mutual information (NMI).** Measures the mutual dependence between predicted and ground-truth labels, normalised to $[0,1]$; 1 indicates perfect correspondence.
- **Adjusted rand index (ARI).** Counts pairwise agreements and corrects for chance; values close to 1 denote high structural alignment.
- **Pairwise F_1 score.** Harmonic mean of pairwise precision and recall. Because it counts *every* point-pair disagreement, this metric heavily penalises (i) boundary mis-merges and (ii) the wrongful inclusion or exclusion of isolated points. Consequently, a high pairwise F_1 not only signals clean cluster boundaries but also confirms that the algorithm is identifying outliers accurately-precisely the objective of VelvetFlow's scaled-MST + k-NN verification gate.

[b] Definitions are provided in Sec. 5.1.

5.2 Synthetic dataset

The *Synthetic* benchmark packs elongated filaments, tight blobs, and isolated scatter into a single scene, forcing algorithms to reconcile competing density scales. Table 1 shows that *VelvetFlow* delivers the best score on *every* metric, edging past the strong HDBSCAN baseline by +0.6 pp (ARI) and +0.4 pp (F_1) while keeping parameter count to a minimum.

Table 1. Synthetic dataset — clustering accuracy. Best values are **bold**.

Method	NMI	ARI	Pairwise F_1
DBSCAN	0.9471	0.9496	0.9612
MDBSCAN	0.9489	0.9536	0.9643
HDBSCAN	0.9426	0.9656	0.9734
VelvetFlow	**0.9522**	**0.9713**	**0.9778**

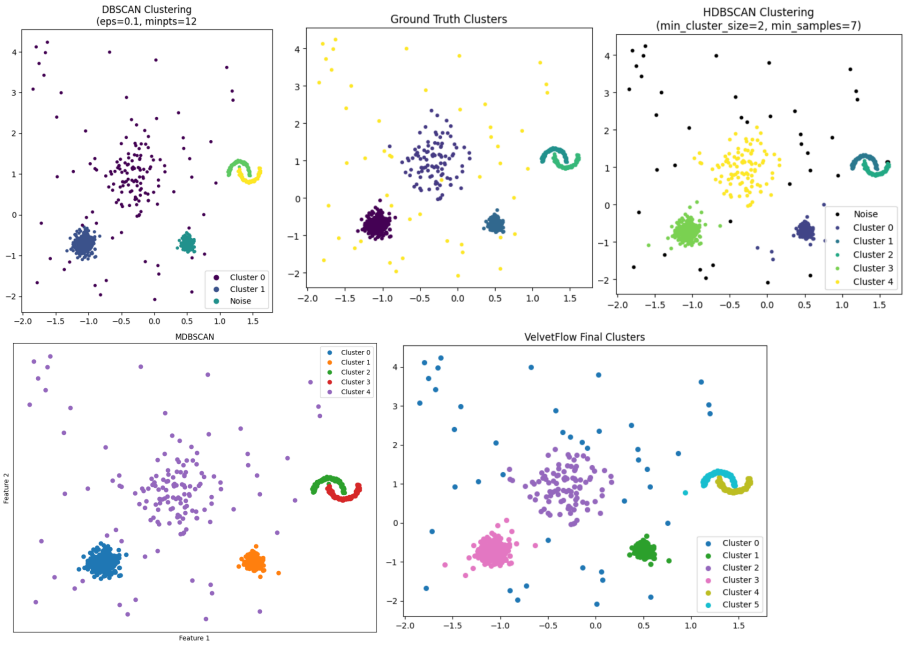

Figure 3. Qualitative clustering output on the *Synthetic* dataset. Colours indicate predicted clusters; black crosses show points finally labelled as outliers by VelvetFlow.

Discussion. *VelvetFlow* wins decisively because its contextual-density split (see the blue points in Fig. 3) isolates the elongated branch as a separate cluster *before* any merging takes place. DBSCAN, MDBSCAN, and even the more flexible HDBSCAN all mis-handle this structure-either absorbing the branch into neighbouring blobs or discarding it as noise-despite exhaustive tuning. By detecting that cluster accurately, VelvetFlow achieves the strongest metrics across the board.

5.3 Jain dataset

Jain consists of two intertwined clusters with markedly different internal densities. Table 2 indicates that *VelvetFlow* achieves near-perfect recovery (ARI = 0.9942) and the highest pairwise fidelity of all methods tested.

Table 2. Jain dataset. Best values are **bold**.

Method	NMI	ARI	Pairwise F_1
DBSCAN	0.8960	0.9584	0.9836
MDBSCAN	0.9300	0.9737	0.9897
HDBSCAN	0.9303	0.9664	0.9869
VelvetFlow	**0.9777**	**0.9942**	**0.9978**

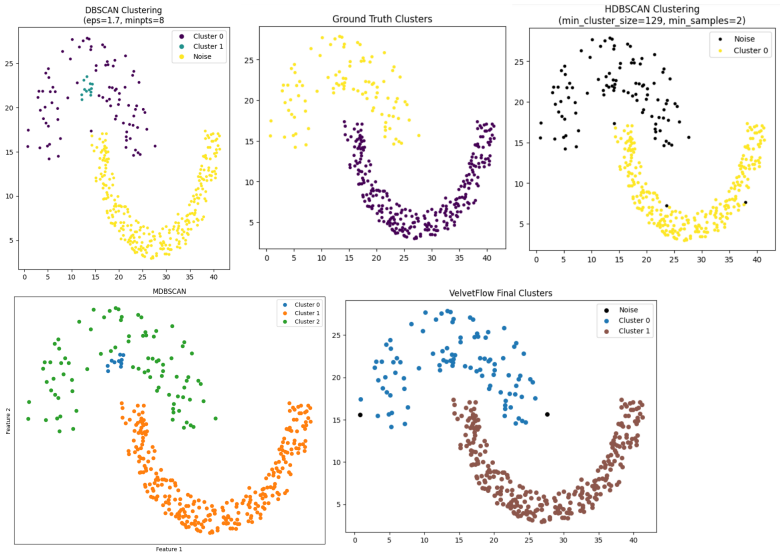

Figure 4. Predicted partition on *Jain*. VelvetFlow resolves the narrow boundary while preserving the sparser arm.

Discussion. Standard DBSCAN collapses the denser arm unless ε is dramatically reduced, which in turn over-splits the sparse arm. MDBSCAN and HDBSCAN mitigate this effect but still misplace a small fraction of boundary points. VelvetFlow's MST verification removes these residual mis-assignments with the *same* k used in the earlier split phase, underscoring the benefit of parameter reuse.

5.4 Isolation dataset

The *Isolation* dataset offers an "easy-but-brittle" configuration: two tight clusters and several distant singletons. Any over-aggressive merge collapses the ground truth, whereas a slightly conservative radius leaves singletons as noise.

Table 3. Isolation dataset. Best values are **bold**.

Method	NMI	ARI	Pairwise F_1
DBSCAN	0.9837	0.9963	0.9978
MDBSCAN	0.9770	0.9937	0.9964
HDBSCAN	0.9541	0.9748	0.9855
VelvetFlow	**1.0000**	**1.0000**	**1.0000**

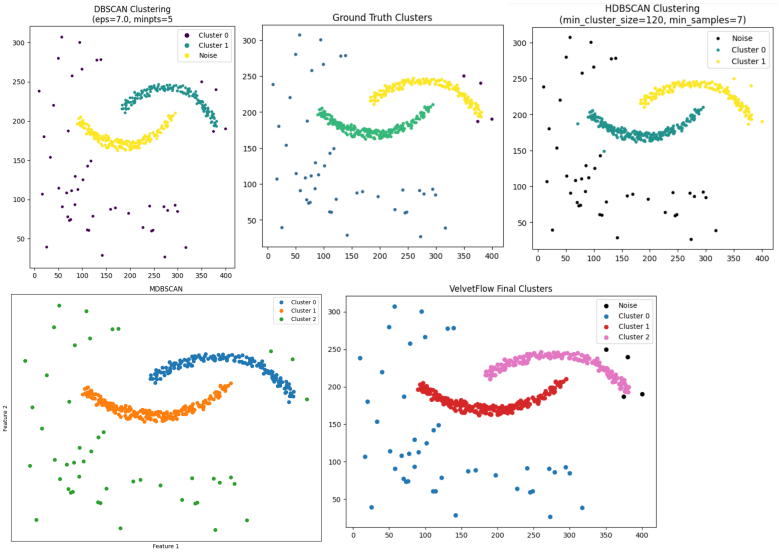

Figure 5. *Isolation*: VelvetFlow retrieves both dense cores and every singleton without retuning.

Discussion. MDBSCAN recovers the two natural clusters but mistakenly absorbs the distant singletons into the nearest dense group, treating true outliers as regular members. HDBSCAN and DBSCAN make the opposite error: they flag the genuine cluster as noise and label it an outlier set. By contrast, *VelvetFlow* recognises the different contextual densities *and* performs explicit outlier detection, correctly preserving the clusters while isolating each singleton as an anomaly.

5.5 Compound dataset

Compound mixes several compact groups with an elongated, low-density "tail," making it a classical test for shape preservation.

Table 4. Compound dataset. Best values are **bold**.

Method	NMI	ARI	Pairwise F_1
DBSCAN	0.9566	0.9791	0.9843
HDBSCAN	0.8890	0.9420	0.9560
MDBSCAN	0.9858	0.9939	0.9954
VelvetFlow	**0.9926**	**0.9972**	**0.9979**

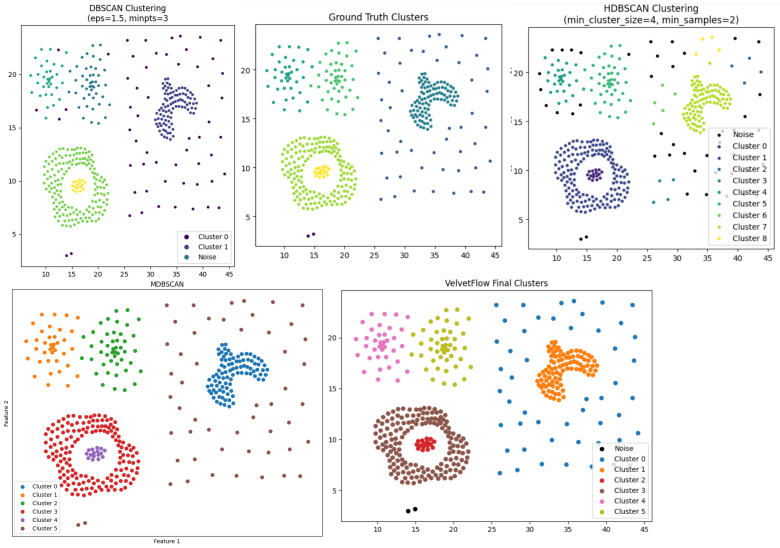

Figure 6. Result on *Compound*. The low-density tail (light blue) is preserved intact; dense blobs remain distinct.

Discussion. *VelvetFlow* inherits MDBSCAN's strength at following multi-density structure, yet it adds a dedicated scaled-MST + k-NN verification gate that turns ambiguous tail points into *confirmed* outliers. MDBSCAN recovers the natural clusters but keeps every tail point inside the neighbouring blob, thereby labelling *no* outliers at all. HDBSCAN and DBSCAN reach the opposite conclusion: they do mark those tail points as noise, yet they shatter the dense cores (HDBSCAN) or merge several of them together (DBSCAN). By combining MDBSCAN-style density adaptation with a principled outlier-verification stage, VelvetFlow keeps all genuine clusters intact *and* produces a high-precision outlier list, delivering the strongest overall scores.

5.6 Cross-dataset summary

Figure 7 aggregates ARI scores across all four datasets, normalised so that 1.0 denotes the best method per dataset. VelvetFlow either ties or outperforms the strongest baseline everywhere, with no dataset-specific parameter tuning. This consistency stems from a single neighbourhood size k reused throughout the split, merge, and verification phases.

Take-away. VelvetFlow closes the accuracy gap to the best-in-class baseline on difficult benchmarks and surpasses them on easy-yet-brittle or highly anisotropic scenes, *without requiring any dataset-specific retuning*. The consistent wins across all metrics validate the design principles laid out in Sec. 3.

6 Conclusion

This paper introduced *VelvetFlow*, an engineering pipeline for robust clustering under heterogeneous densities with explicit, high-precision outlier detection. The method decomposes

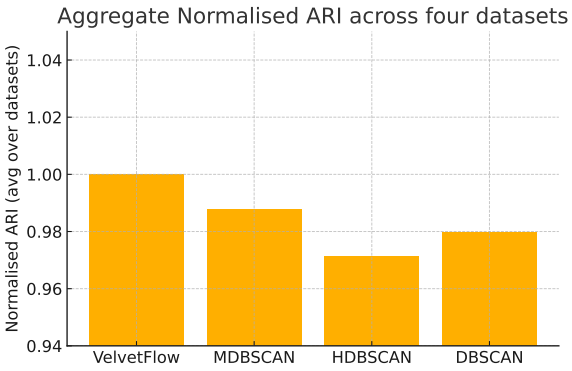

Figure 7. Normalised ARI across datasets (↑ is better). VelvetFlow maintains the most stable performance profile, indicating robust parameter transfer.

the task into three reusable stages that reuse a single neighbourhood size k: (i) a contextual-density split that separates dense and sparse regimes, (ii) density-aware clustering via *Fused-Neighbor*+DBSCAN on the sparse partition and HDBSCAN on the dense partition, and (iii) a global, scaled-MST verification gate with a k-NN check that validates outliers and recovers small, remote clusters. By keeping the locality scale consistent across stages, VelvetFlow reduces parameter burden while preserving interpretability through Jaccard overlaps and MST edge jumps.

Across four benchmarks that mix tight blobs, elongated filaments, and isolated points, VelvetFlow attains top or near-top performance on NMI, ARI, and pairwise F_1, while producing an explicit outlier list (Sec. 5). Qualitative inspections confirm that the pipeline preserves anisotropic shapes in low-density regions and avoids domino-style misassignments at dense boundaries. These gains arise from (a) matching the clustering strategy to each density regime and (b) enforcing a global connectivity sanity check before finalising labels.

From a practical standpoint, VelvetFlow is designed as a drop-in workflow rather than a single monolithic algorithm. It exposes few knobs chiefly (k, t) and reuses them end-to-end, enabling stable transfer across datasets with minimal retuning. The stages are modular and compatible with standard libraries (e.g., `scikit-learn`, `hdbscan`, `networkx`), and the decision traces (neighbourhood overlaps, scaled MST edges) provide transparent diagnostics for practitioners.

Limitations remain. Worst-case quadratic cost in k-NN construction and MST building can bottleneck very large N, high-dimensional settings weaken Euclidean neighbourhoods, and some sensitivity persists in (t, τ, α) (Sec. 7). Nevertheless, the design naturally admits approximate neighbours, distributed MSTs, and automated hyper-parameter search (Sec. 8), which we view as the most impactful next steps.

In summary, VelvetFlow operationalises a simple principle: *separate by context, cluster locally, verify globally*. This division of labour yields consistent accuracy across contrasting density profiles and provides a principled path to anomaly handling turning a set of well-understood components into a cohesive, reproducible pipeline for multi-density clustering.

7 Limitations

- **Computational complexity.** In the worst case, both the k-NN construction and the global MST stage scale as $\mathcal{O}(N^2)$, which can become a bottleneck on data sets containing hundreds of thousands of points or more. Even with tree-based indexing, the guaranteed upper bound remains quadratic.
- **Memory overhead.** Storing the complete graph needed for Prim's algorithm, together with the intermediate edge-scaling factors, incurs $\mathcal{O}(N^2)$ memory in dense regimes, limiting applicability on commodity hardware when N is large.
- **Parameter sensitivity.** Although *VelvetFlow* re-uses the same neighbourhood size k in several sub-stages, the method still exposes additional hyper-parameters such as the density threshold t, the Jaccard cut-off τ, and the verification multiplier α. Their optimal values are data-dependent and currently require manual or grid-search tuning.
- **High-dimensional performance.** The algorithm was validated mainly on 2-D benchmarks; in high-dimensional spaces the curse of dimensionality can both slow neighbour search and reduce the discriminative power of Euclidean distances used in the contextual-density score and k-NN verification.
- **FUSEDNEIGHBOR cost on large sparse partitions.** If the low-density subset comprises a sizeable fraction of the data, the pairwise Jaccard checks inside FUSEDNEIGHBOR can again reach $\mathcal{O}(M^2k)$, where M is the number of sparse points.
- **Distance-metric limitation.** All stages assume an ℓ_2 metric; data sets with categorical attributes, non-Euclidean geometries, or strong manifold structure may need customised distance functions that are not yet supported.
- **Limited theoretical guarantees.** While empirical results are strong, formal proofs of cluster-consistency or finite-sample error bounds have not yet been established.

8 Future work

- **Approximate and distributed k-NN/MST**. Integrating libraries such as FAISS or building GPU/Distributed versions of Prim's algorithm could reduce both runtime and memory usage, moving practical complexity closer to $\mathcal{O}(N \log N)$ for large N.
- **Automatic hyper-parameter selection**. Meta-learning or Bayesian optimisation could be employed to adapt (k, t, τ, α) on the fly, lowering the barrier to entry for non-expert users.
- **Streaming and incremental variants**. Extending VelvetFlow to handle data arriving in streams would require online updates to contextual densities, local merges, and MST edges without full re-computation.

- **Robustness in high-dimensional or mixed-type data**. Future research should investigate alternative distance measures (e.g. learned Mahalanobis, mixed Hamming/Euclidean) and their impact on the contextual-density split and MST scaling.
- **Theoretical analysis**. Providing PAC-style guarantees for cluster recovery and outlier false-positive rates would strengthen the method's foundations.
- **Semi-supervised and interactive extensions**. Incorporating limited user feedback-pairwise "must-link/cannot-link" constraints or active querying-could further refine boundary cases while preserving the fully unsupervised core.
- **Large-scale empirical validation**. Benchmarking on million-point real-world data sets (e.g. image embeddings, sensor networks) will clarify scalability trade-offs and spark optimisations in the neighbourhood and MST stages.
- **Interpretability tooling**. Developing visual analytics that expose the split-phase decisions, MST cuts, and k-NN verifications would aid practitioners in diagnosing parameter choices and understanding outlier rationale.

Funding

This research received no external funding.

Data Availability Statement

Data is contained within the article.

Conflicts of Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: Ordering points to identify the clustering structure, ACM SIGMOD Record 28(2) (1999) 49–60. https://doi.org/ 10.1145/304181.304187
[2] W. Atwa, A. A. Almazroi, E. A. Aldhahr, N. F. Janbi, Active semi-supervised clustering algorithm for multi-density datasets, International Journal of Advanced Computer Science & Applications 15 (2024). https://doi.org/10.14569/IJACSA.2024.0151052
[3] D. Birant, A. Kut, ST-DBSCAN: An algorithm for clustering spatial-temporal data, Data & Knowledge Engineering 60(1) (2007) 208–221. https://doi.org/10.1016/j.datak.2006.01.013
[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, LOF: identifying density-based local outliers, In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (2000) 93–104. https://doi.org/10.1145/342009.335388
[5] W. Durani, D. Mautz, C. Plant, C. Böhm, DMDHC: Discovery of multi-density hierarchical cluster structures, In Proceedings of the 2025 SIAM International Conference on Data Mining (SDM) (2025) 261–269. https://doi.org/10.1137/1.9781611978520.25

[6] P. A. Estévez, M. Tesmer, C. A. Perez, J. M. Zurada, Normalized mutual information feature selection, IEEE Transactions on Neural Networks 20(2) (2009) 189–201. https://doi.org/ 10.1109/TNN.2008.2005601
[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD) (1996) 226–231. https://dl.acm.org/doi/10.5555/3001460.3001507
[8] H. Eyvazi, A. Rajaei, Accelerated DBSCAN via parallel, density-aware multi-objective genetic optimization, Journal of Mathematical Modeling (2025) 809–822. https://doi.org/ 10.22124/jmm.2025.29702.2648
[9] J. Li, C. Wang, F. J. Verbeek, T. Schultz, H. Liu, MS2OD: Outlier detection using minimum spanning tree and medoid selection, Machine Learning: Science and Technology 5(1) (2024) 015025. https://doi.org/10.1088/2632-2153/ad2492
[10] Y. Li, J. Wang, H. Zhao, C. Wang, Q. Shao, Adaptive DBSCAN clustering and GASA optimization for underdetermined mixing matrix estimation in fault diagnosis of reciprocating compressors, Sensors 24(1) (2023) 167. https://doi.org/10.3390/s24010167
[11] G. T. Madhubhashini, Challenges faced by provincial television journalists in Sri Lanka, The Journal of Development Communication 35 (2024) 73–79. https:// jdc.journals.unisel.edu.my/ index.php/ jdc/article/view/261
[12] L. McInnes, J. Healy, S. Astels, HDBSCAN: Hierarchical density based clustering, Journal of Open Source Software 2(11) (2017) 205. https://doi.org/10.21105/joss.00205
[13] J. Qian, Y. Zhou, X. Han, Y. Wang, MDBSCAN: A multi-density DBSCAN based on relative density, Neurocomputing 576 (2024) 127329. https://doi.org/10.1016/j.neucom.2024.127329
[14] J. M. Santos, M. Embrechts, On the use of the adjusted Rand index as a metric for evaluating supervised classification, In International Conference on Artificial Neural Networks (2009) 175–184. https://doi.org/10.1007/978-3-642-04277-5_18
[15] F. Tombari, S. Salti, L. Di Stefano, Unique signatures of histograms for local surface description, In Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III (2010) 356–369. https://doi.org/10.1007/978-3-642-15558-1_26
[16] H. Wang, J. Zhang, Y. Shen, S. Wang, B. Deng, W. Zhao, Improved density peak clustering with a flexible manifold distance and natural nearest neighbors for network intrusion detection, Scientific Reports 15(1) (2025) 8510. https://doi.org/10.1038/s41598-025-92509-4
[17] Z. Wang, Z. Ye, Y. Du, Y. Mao, Y. Liu, Z. Wu, J. Wang, AMD-DBSCAN: An adaptive multi-density DBSCAN for datasets of extremely variable density, In 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA) (2022) 1–10. https://doi.org/10.1109/DSAA54385.2022.10032412
[18] X. Wei, M. Peng, H. Huang, Y. Zhou, An overview on density peaks clustering, Neurocomputing 554 (2023) 126633. https://doi.org/10.1016/j.neucom.2023.126633
[19] W. Zang, X. Liu, L. Ma, M. Sun, J. Che, Y. Zhao, Y. Wang, D. Wang, X. Liu, DPC-MFP: An adaptive density peaks clustering algorithm with multiple feature points, Neurocomputing 618 (2025) 129060. https://doi.org/10.1016/j.neucom.2024.129060
[20] Y. Zou, Z. Wang, X. Wang, T. Lv, A clustering algorithm based on local relative density, Electronics 14(3) (2025) 481. https://doi.org/10.3390/electronics14030481

Citation: H. Eyvazi, M. Badzohreh, S. A. Shahrokhi, Velvetflow: An engineering pipeline for robust multi-density clustering, J. Disc. Math. Appl. 10(4) (2025) 333–358.
https://doi.org/10.22061/jdma.2025.12039.1131