Data Mining Project: Clustering on Mall Customer Dataset

Daoyuan Guo Jinghong Tan Guofu Tang

December 2025

Abstract

Customer segmentation is a core task in retail analytics. In this project we build an end-to-end clustering pipeline on the *Mall Customer Dataset*, a small but clean tabular dataset from Kaggle with 200 customers and five attributes. The pipeline covers data auditing, outlier detection, feature engineering, multi-stage feature selection, model comparison across K-Medoids, Gaussian Mixture Models (GMM) and HDBSCAN, and a final business-oriented segmentation based on HDBSCAN. In addition, we implement LLM-assisted KMeans and GMM pipelines as AI-generated baselines, and evaluate their stability via the Adjusted Rand Index (ARI). We present quantitative results using internal validation indices (Silhouette, Calinski–Harabasz, Davies–Bouldin) and a set of visualizations, and we discuss the strengths, limitations and possible extensions of the proposed approach.

1 Introduction

Customer segmentation aims to partition a customer base into homogeneous groups that differ in demographic structure, spending behavior and responsiveness to marketing actions. For malls and retailers, a good segmentation allows differentiated pricing, targeted promotions and more efficient loyalty programs.

We use the *Mall Customer Dataset* [6], a widely used toy dataset from Kaggle. It contains 200 rows and five fields: `CustomerID`, `Genre`, `Age`, `Annual Income (k$)`, and `Spending Score (1--100)`. The small scale and clean schema make it ideal for exploring classical clustering methods, internal validation indices and low-dimensional visualizations.

The project is organized according to the course specification:

- **Task A** (Data acquisition, understanding and processing): schema audit, basic statistics, outlier detection, feature engineering and feature selection.
- **Task B** (Clustering with visualization): implementation of multiple clustering algorithms, hyperparameter tuning via internal metrics, 2D visualizations (t-SNE / PCA), final model selection and human-readable labels.
- **Task C** (Modeling using AI-related tools): LLM-assisted implementations of KMeans and GMM, stability analysis across seeds, and comparison with our hand-crafted pipeline.

This report is structured as follows. Section 2 describes the methodology, including data preprocessing, feature engineering, clustering models and AI-assisted baselines. Section 3 details the experimental setup and hyperparameter grids. Section 4 presents the quantitative and visual results with analysis. Section 5 summarises the main findings, and Section 6 discusses limitations and possible improvements. The mathematical definitions of internal indices and algorithms used for outlier detection and model selection are provided in the Appendix.

2 Methodology

2.1 Dataset understanding and schema

We start from the CSV file provided by Kaggle. A basic inspection using `pandas.DataFrame.info()` returns the overview in Table 1.

Table 1: Dataset overview: Mall customer records

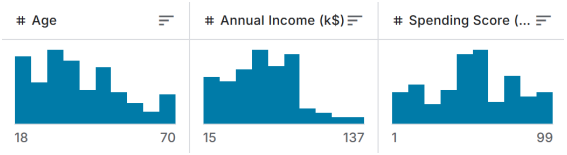
Column #	Column Name	Non-Null Count	Data Type
0	CustomerID	200 non-null	int64
1	Genre	200 non-null	object
2	Age	200 non-null	int64
3	Annual Income (k$)	200 non-null	int64
4	Spending Score (1-100)	200 non-null	int64

Additional dataset info:
RangeIndex: 200 entries (0 to 199)
Data types: int64 (4 columns), object (1 column)
Memory usage: 7.9+ KB

In addition, we manually inspect the dataset schema and distributional properties. Figure 1 shows the schema metadata view and the empirical distributions.

	name	dtype	non_null	nulls	unique	examples	min	max	mean	memory_bytes
0	CustomerID	int64	200	0	200	[1, 2, 3]	1.0	200.0	100.50	1732
1	Genre	object	200	0	2	[Female, Male]	NaN	NaN	NaN	10956
2	Age	int64	200	0	51	[32, 35, 31]	18.0	70.0	38.85	1732
3	Annual Income (k$)	int64	200	0	64	[54, 78, 60]	15.0	137.0	60.56	1732
4	Spending Score (1-100)	int64	200	0	84	[42, 55, 73]	1.0	99.0	50.20	1732

(a) Manual metadata retrieval.



(b) Data distribution.

Figure 1: Schema details and empirical distributions from the Mall Customer Dataset.

According to the Kaggle description and our checks:

- The gender distribution is 56% female and 44% male.
- Age and annual income show right-skewed distributions.
- The spending score displays a three-peak structure reminiscent of a mixture distribution.

2.2 Preprocessing and outlier detection

Basic cleaning. We drop the purely identificational column `CustomerID`. The categorical feature `Genre` is encoded as a binary indicator (`Genre_encoded` = 1 for female, 0 for male). No missing values are present. We check for duplicate rows (after removing `CustomerID`) and find none.

An example of the cleaned dataset head is shown in Figure 2.

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	21	15	81
1	0	23	16	77
2	0	31	17	40
3	0	22	17	76
4	0	35	18	6

Figure 2: Example of cleaned data (after removing `CustomerID` and encoding `Genre`).

Outlier detection. To flag potentially anomalous records we apply two unsupervised detectors on the numeric features with contamination rate 0.5:

- **Isolation Forest** (see Appendix A.7).
- **Local Outlier Factor (LOF)** (see Appendix A.6).

Each method marks 10 points as outliers; their intersection (7 customers) is treated as high-confidence anomalies. Instead of removing them, we retain these points and rely on HDBSCAN's explicit noise label for downstream handling.

All numerical features are standardised to zero mean and unit variance before clustering and dimensionality reduction.

2.3 Feature engineering

The original 3 numeric attributes (`Age`, `Annual Income`, `Spending Score`) are insufficient to capture nuanced structure, so we design a richer feature set.

Constructed features. We create:

1. **Polynomial and interaction terms (degree 2):**
   - Squared terms: `Age`$\hat{2}$, `Annual Income (k$)`$\hat{2}$, `Spending Score (1--100)`$\hat{2}$.
   - Pairwise products: `Age * Income`, `Age * SpendingScore`, `Income * SpendingScore`.
2. **Age binning via KMeans:** we cluster age into 6 groups using 1D KMeans and select $K = 6$ based on the best Silhouette score (≈ 0.6). The resulting age groups are shown in Table 2.
3. **Income binning via quartiles:** we discretise annual income into four quartile-based tiers `IncomeBin_Q1`–`Q4`.
4. **Ratio features:**
   - `Income_per_Age` = income / age.
   - `Spend_to_Income` = spending score / income.

Table 2: Age binning result (Task A)

Age group	17–24	25–32	33–41	42–51	52–60	61–69
Count	36	43	42	38	19	15

These engineered features improve downstream predictiveness: a logistic regression predicting gender increases from accuracy 0.5751 / ROC AUC 0.5016 (original features) to accuracy 0.6788 / ROC AUC 0.6458 after feature engineering.

Filter-based feature selection. We start from the full set of numeric columns, including engineered terms and dummies, as summarised in Table 3. We then apply:

- a variance filter to drop constant or near-constant features; and
- a correlation filter to drop features with absolute Pearson correlation > 0.9 with another feature.

The correlation matrix is visualised in Figure 3.

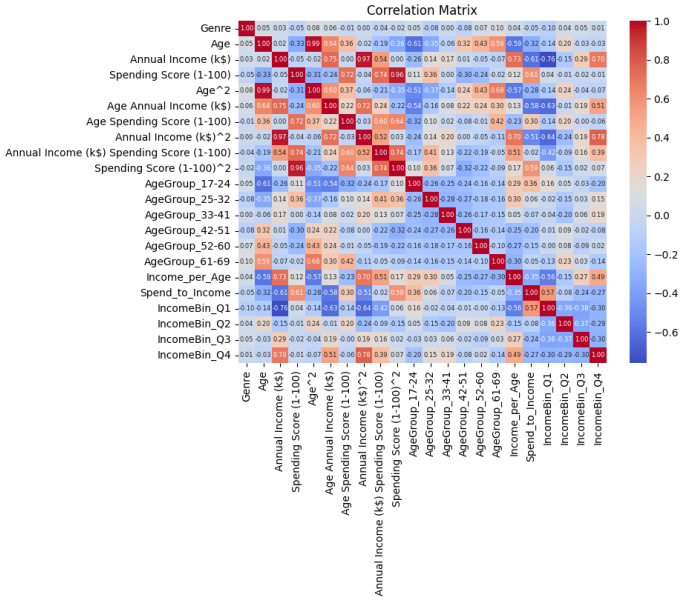


Figure 3: Correlation matrix of processed features.

Wrapper and embedded selection. To further reduce redundancy and focus on clustering-relevant variables, we proceed in two steps:

- **Wrapper stage:** we run KMeans with $K = 7$ on different subsets and keep subsets with high Silhouette scores. One such subset (Silhouette = 0.5193) is:

  `['IncomeBin_Q4', 'IncomeBin_Q1', 'Genre', 'AgeGroup_61-69', 'AgeGroup_52-60', 'Annual Income (k$)', 'IncomeBin_Q3', 'IncomeBin_Q2', 'Age Annual Income (k$)', 'Income_per_Age']`.

Table 3: Feature selection process for customer data (filter stages)

Stage	**Feature list (summary)**
Numeric columns (initial)	Genre, Age, Annual Income, Spending Score, Age$\hat{2}$, Age*Income, Age*Score, Income$\hat{2}$, Income*Score, Score$\hat{2}$, Age-group dummies, Income_per_Age, Spend_to_Income, Income-bin dummies
Kept after variance filter	All except zero-variance features (none in practice)
Dropped (high correlation > 0.9)	Age$\hat{2}$, Income$\hat{2}$, Score$\hat{2}$
Kept after correlation filter (final filter stage)	Genre, Age, Income, Score, Age*Income, Age*Score, Income*Score, Age-group dummies, Income_per_Age, Spend_to_Income, Income-bin dummies

- **Embedded stage:** on a wrapper-selected subset, we train a Random Forest classifier to predict the KMeans labels and rank features by importance. The importances are reported in Table 4.

Table 4: Random Forest feature importances (descending order)

Feature	Importance	Feature	Importance	Feature	Importance
Spending Score (1–100)	0.1264	Annual Income (k$)	0.1244	Income*Score	0.1132
Spend_to_Income	0.0938	Age	0.0906	Age*Income	0.0880
IncomeBin_Q1	0.0778	Income_per_Age	0.0689	Age*Score	0.0644
AgeGroup_17-24	0.0506	AgeGroup_61-69	0.0408	IncomeBin_Q2	0.0308
IncomeBin_Q3	0.0067	AgeGroup_25-32	0.0066	IncomeBin_Q4	0.0059
AgeGroup_42-51	0.0058	AgeGroup_33-41	0.0023	AgeGroup_52-60	0.0018
Genre	0.0012				

Finally, we obtain a compact clustering-ready feature set (7 core features) via majority vote between wrapper-selected and importance-ranked features. The fully preprocessed dataset is previewed in Figure 4.

	Age Annual Income (k$)	AgeGroup_61-69	Annual Income (k$)	IncomeBin_Q1	IncomeBin_Q2	IncomeBin_Q3	Income_per_Age
0	-1.699593	-0.290292	-1.866777	1.646675	-0.591312	-0.623289	-1.137799
1	-1.655382	-0.290292	-1.825592	1.646675	-0.591312	-0.623289	-1.158515
2	-1.522749	-0.290292	-1.784407	1.646675	-0.591312	-0.623289	-1.322240
3	-1.650377	-0.290292	-1.784407	1.646675	-0.591312	-0.623289	-1.072825
4	-1.436830	-0.290292	-1.743222	1.646675	-0.591312	-0.623289	-1.360152

Figure 4: Preview of preprocessed, standardised feature matrix.

2.4 Clustering models and internal metrics

We implement three main clustering models in Task B:

K-Medoids. A prototype-based algorithm similar to KMeans, but using medoids (actual points) as cluster centers. This improves robustness to outliers and allows general distance metrics.

Gaussian Mixture Model (GMM). A parametric model-based clustering method that assumes data is generated from a mixture of Gaussians with soft assignments. We consider covariance structures `full`, `tied`, `diag` and `spherical` and use information criteria (AIC, BIC; Appendix A.1) plus internal indices to select hyperparameters.

HDBSCAN. A density-based algorithm that builds a cluster hierarchy from a minimum spanning tree in a transformed space and extracts stable clusters while labelling low-density points as noise. Intuition and details are summarised in Appendix A.5.

For internal validation we use:

- **Silhouette score** (cluster separation vs cohesion).
- **Calinski–Harabasz (CH) index** (between-/within-cluster dispersion ratio; Appendix A.3).
- **Davies–Bouldin (DB) index** (average worst-case cluster similarity; Appendix A.4).

2.5 Visualization and segment labelling

To understand the structure of the feature space and communicate results, we use:

- **t-SNE** on the full standardised feature matrix to obtain 2D embeddings, and color points by cluster labels from K-Medoids, GMM and HDBSCAN.
- **Correlation analysis** between t-SNE axes and the original engineered features to interpret main directions.

Figure 5 shows the t-SNE visualizations under each model.

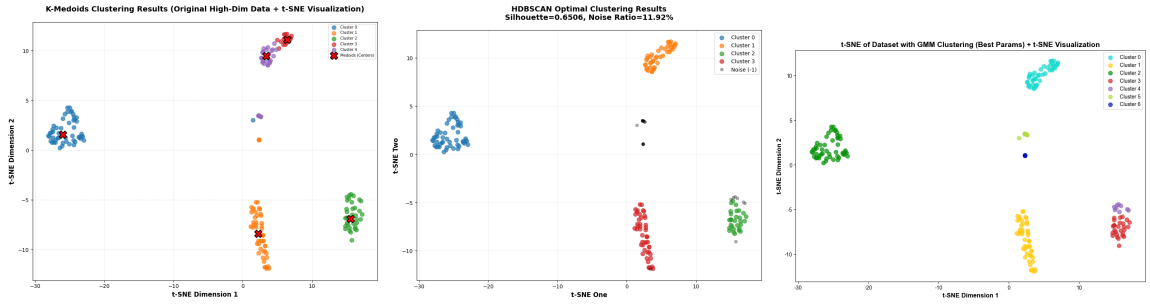


Figure 5: t-SNE embedding colored by K-Medoids labels (left), HDBSCAN labels (middle) and GMM labels (right).

We also compute Pearson correlations between key features and the t-SNE axes; for example:

Feature	Corr(t-SNE_1)	Corr(t-SNE_2)
Annual Income (k$)	0.875	-0.558
IncomeBin_Q1	-0.937	0.210
IncomeBin_Q2	0.242	0.781
Income_per_Age	0.650	-0.494

This confirms that t-SNE_1 primarily reflects income and income tiers, while t-SNE_2 emphasises differences in mid vs higher income and income-per-age ratios.

For the final HDBSCAN solution, we map numeric labels to semantic segments (Low / Mid-Low / Mid-High / High Income + Outliers) and display them in the 2D embedding (Figure 6) and as distribution plots by segment (Figures 7 and 8).

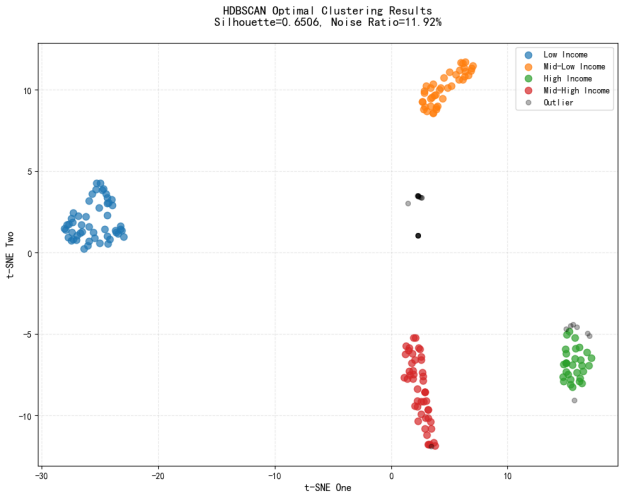


Figure 6: t-SNE visualization of HDBSCAN clusters with human-readable segment labels.

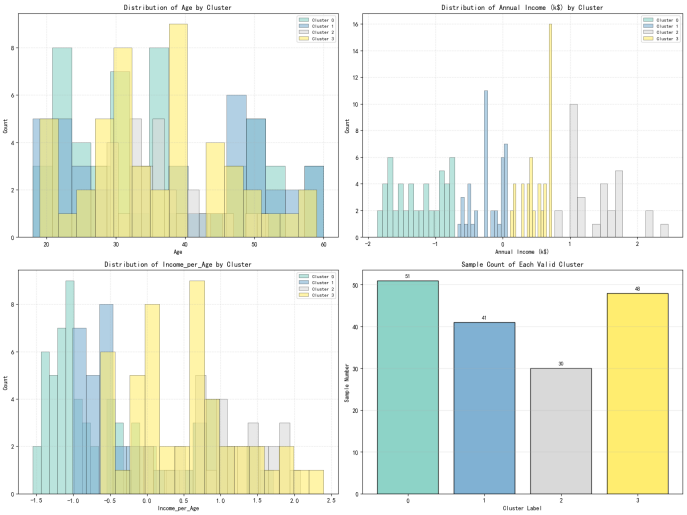


Figure 7: Univariate distributions (Age, Income, Income_per_Age) by HDBSCAN segment.

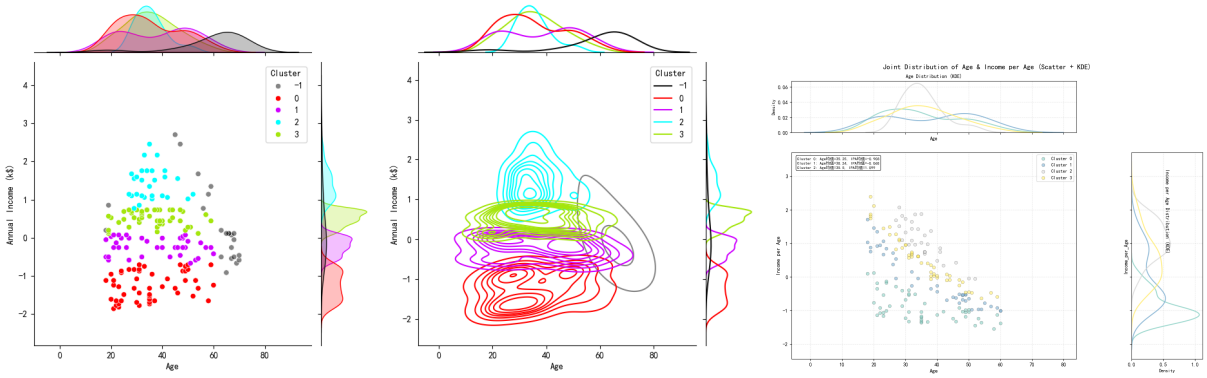


Figure 8: Age–income scatter (left), KDE distribution (middle) and Age vs Income_per_Age scatter (right) colored by HDBSCAN segment.

2.6 AI-related tools: LLM-generated baselines

For Task C, we ask a large language model to generate Python code that:

- loads and standardises the same dataset,
- runs KMeans (with KMeans++ initialization) and GMM,
- selects k / $n_{\text{components}}$ using elbow and Silhouette methods (KMeans) or AIC/BIC (GMM),
- visualises clusters using PCA.

The LLM-generated KMeans and GMM models are evaluated on the same features using the same internal metrics. The resulting PCA plots are shown in Figure 9, and hyperparameter selection curves in Figure 10.

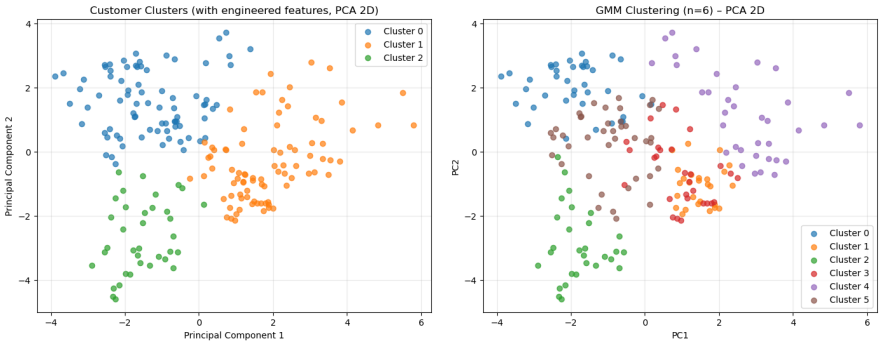


Figure 9: AI-generated KMeans (left) and GMM (right) segments projected onto first two principal components.

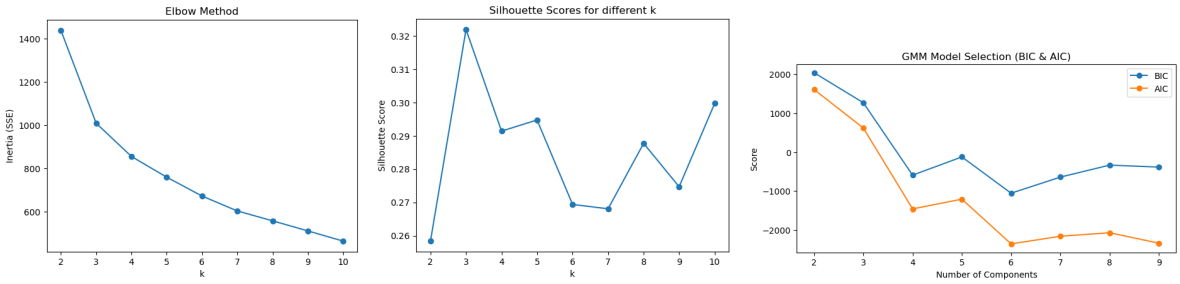


Figure 10: Elbow curve (left) and Silhouette scores (middle) for KMeans; AIC and BIC vs number of components for GMM (right).

The internal metrics for these AI-generated models are summarised in Table 5.

Table 5: AI-generated clustering evaluation metrics (Task C)

Method	Silhouette	CH	DB
KMeans	**0.3219**	**96.81**	**1.1222**
GMM	0.2201	65.48	1.3249

We further assess stability across seeds via the Adjusted Rand Index (Appendix A.2). Summary statistics are given later in Table 7.

3 Description of Experiments

3.1 Implementation environment

All experiments are implemented in Python using:

- `pandas`, `numpy` for data wrangling.
- `scikit-learn` for KMeans, GMM, Random Forest, Isolation Forest, LOF and metrics.
- `hdbscan` for HDBSCAN.
- `scikit-learn-extra` (or equivalent) for K-Medoids.

Because clustering is unsupervised, no train/test split is used; instead, we rely on internal criteria and stability analyses.

3.2 Hyperparameter grids

K-Medoids. We explore:

$$K \in \{3,4,5,6,7,8,9\}, \quad \text{max\_iter} \in \{300,500,800\}, \quad \text{metric} \in \{\text{Euclidean}, \text{Manhattan}\}.$$

Silhouette and CH scores as a function of K are reported in Table 6 and visualised in Figure 11.

Table 6: K-Medoids: best metrics per number of clusters

K	Best Silhouette	Best CH index
3	0.5136	141.61
4	0.5535	154.82
5	0.5279	143.00
6	0.5058	125.54
7	**0.6002**	229.75
8	0.5771	**247.85**
9	0.4941	239.77

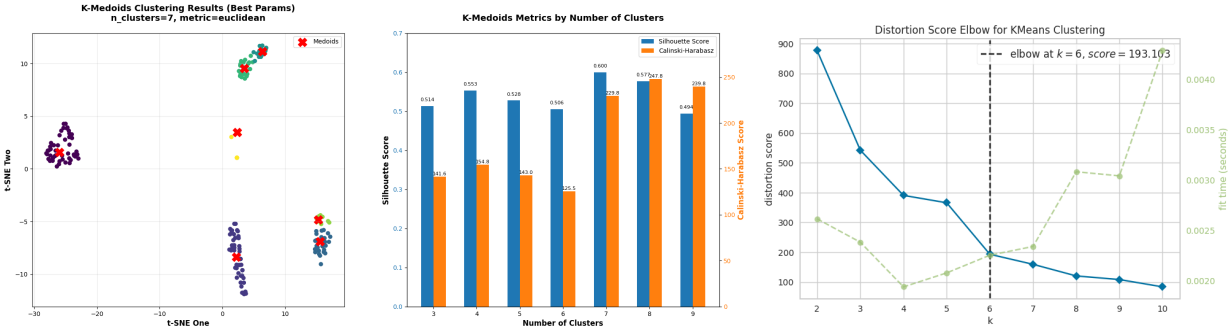


Figure 11: Left: K-Medoids metrics (Silhouette and CH) by number of clusters. Right: inertia curve from KMeans used as a proxy for identifying reasonable K values.

We adopt $K = 7$ as K-Medoids' best configuration based on the trade-off between Silhouette and CH and visual inspection of inertia.

GMM. For GMM we perform a BIC-guided two-stage selection:

1. Fix covariance type to `full`, scan $n_{\text{components}} \in \{2, \ldots, 10\}$ and inspect BIC; see Figure 12.
2. For each covariance type in {`full`, `tied`, `diag`, `spherical`}, select the n with lowest BIC and then evaluate Silhouette and CH. The metrics by covariance type are shown in Figure 13.

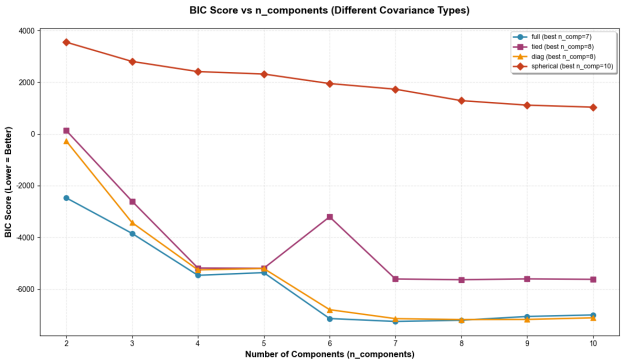


Figure 12: BIC vs number of components for GMM (covariance = full).

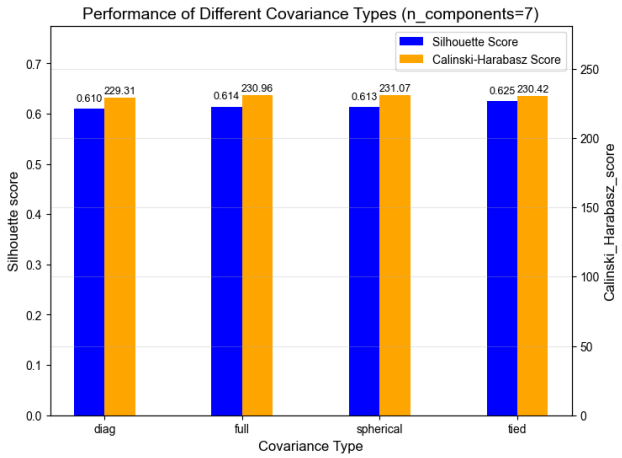


Figure 13: Silhouette and CH indices for GMM under different covariance types and BIC-selected component numbers.

The Silhouette-optimal configuration is $n_{\text{components}} = 7$, covariance type `tied`, with Silhouette $= 0.6249$ and CH $= 230.42$.

HDBSCAN. For HDBSCAN we use the grid:

- `min_cluster_size` $\in \{10, 15, 20, 25\}$,
- `min_samples` $\in \{5, 10, 15\}$ with `min_samples` $\leq$ `min_cluster_size`,
- `cluster_selection_epsilon` $\in \{1.0, 1.5, 2.0\}$,
- metric $\in \{\text{Euclidean}, \text{Manhattan}\}$,
- `cluster_selection_method` $=$ `eom`.

Metrics as a function of parameter combinations are summarised in Figure 14.

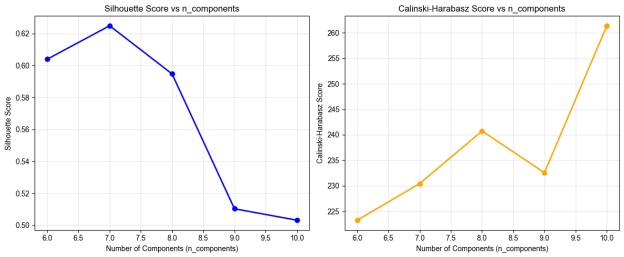


Figure 14: HDBSCAN metrics (Silhouette, CH, noise ratio) across hyperparameter combinations.

The best configuration is:

$$\texttt{min\_cluster\_size} = 15, \quad \texttt{min\_samples} = 15, \quad \varepsilon = 2.0, \quad \text{metric} = \text{Manhattan},$$

which yields Silhouette $= 0.6506$, CH $= 333.34$, noise ratio $\approx 11.9\%$ and four effective clusters.

3.3 Stability experiments for AI-generated models

For the AI-generated KMeans and GMM pipelines, we evaluate stability over multiple random seeds. We run:

- KMeans: 10 runs with different seeds.
- GMM: 20 runs with different seeds.

For each run, we compute the ARI with respect to a reference run (fixed seed). The sample of ARI values (rounded to 3 decimals) is:

$$\begin{aligned}\text{KMeans ARI: } &[0.503, 0.479, 0.500, 0.407, 0.378, 0.391, 0.356, 0.395, 0.426, 0.401],\\ \text{GMM ARI: } &[0.467, 0.442, 0.414, 0.497, 0.424, 0.492, 0.515, 0.494, 0.487, 0.455,\\ &0.534, 0.529, 0.456, 0.471, 0.672, 0.309, 0.590, 0.591, 0.452, 0.448].\end{aligned}$$

Summary statistics are given in Table 7.

Table 7: Stability statistics (ARI across seeds) for AI-generated models

Method	#Runs	Mean ARI	Std ARI	Min / Max
KMeans	10	0.424	**0.049**	**0.356** / 0.503
GMM	20	**0.487**	0.074	0.309 / **0.672**

KMeans is more consistent (lower standard deviation) but has lower mean ARI; GMM achieves higher mean ARI but with larger variability across seeds.

4 Observations and Analysis of Results

4.1 Model comparison and final choice

The best configurations for the three main clustering models are compared in Table 8 and Figure 15.

Table 8: Comparison of optimal models (Task B(a))

Method	Silhouette	Calinski–Harabasz	Davies–Bouldin
K-Medoids	0.6002	229.75	0.6240
GMM	0.6249	230.42	0.5754
HDBSCAN	**0.6506**	**333.34**	**0.5714**

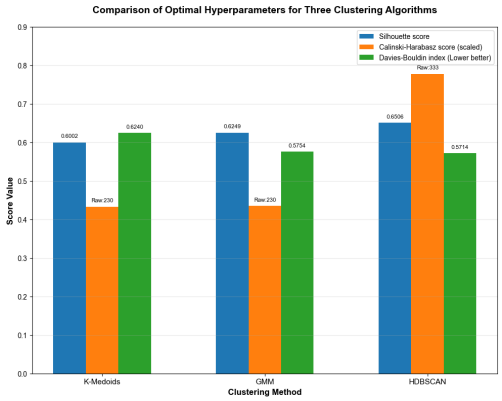


Figure 15: Metric comparison between K-Medoids, GMM and HDBSCAN (Silhouette, CH, DB).

Observation. HDBSCAN dominates on all three internal indices and additionally provides explicit noise handling. This, combined with the interpretable density-based view, makes it the natural final choice for segmentation.

4.2 HDBSCAN segments and marketing interpretation

We map HDBSCAN labels to semantic segments:

$$-1 \to \text{Outlier}, 0 \to \text{Low Income}, 1 \to \text{Mid-Low Income}, 2 \to \text{High Income}, 3 \to \text{Mid-High Income}.$$

The segment footprints in t-SNE space are shown in Figure 6, and the univariate and joint distributions in Figures 7 and 8. Qualitatively:

- **Low Income:** low income and low income-per-age across almost all ages; consistently low spending and low spending-to-income. This is a large but low-value segment, suitable for basic maintenance and low-cost engagement.
- **Mid-Low Income:** slightly higher income and income-per-age; modestly higher spending, with potential to respond to affordable promotions and simple up-selling.
- **Mid-High Income:** solid income and higher income-per-age; moderate to high spending; natural targets for value-added and mid-premium offering strategies.
- **High Income:** highest income and income-per-age; key high-value segment for premium products, personalised services and loyalty programs.
- **Outliers:** atypical age–income combinations and extremely high or low spending intensity; these should be investigated separately either as noise or as niche marketing opportunities.

Overall, the segments form a coherent story: the main separating axes are absolute and relative income, with age acting as a secondary factor for capacity and lifestyle, while spending intensity differentiates high-engagement segments from conservative ones.

4.3 Insights from AI-generated KMeans and GMM

Although the AI-generated models have weaker internal metrics (Table 5), their segment profiles qualitatively agree with the HDBSCAN story.

KMeans segments. Cluster sizes and aggregated statistics are shown in Tables 9–10. The female ratio plot is given in Figure 16.

Table 9: KMeans cluster sizes (AI-generated pipeline)

Cluster	0	1	2
Count	79	85	36

Table 10: KMeans cluster profiles (mean values; AI-generated pipeline)

Cluster	Age	Income (k$)	Score	Spend/Income	Income/Age	Age_group	Income_tier
0	53.14	54.27	37.22	0.73	1.05	2.57	0.86
1	30.54	80.25	56.59	0.75	2.69	0.80	1.40
2	27.11	27.89	63.61	2.56	1.06	0.42	0.08

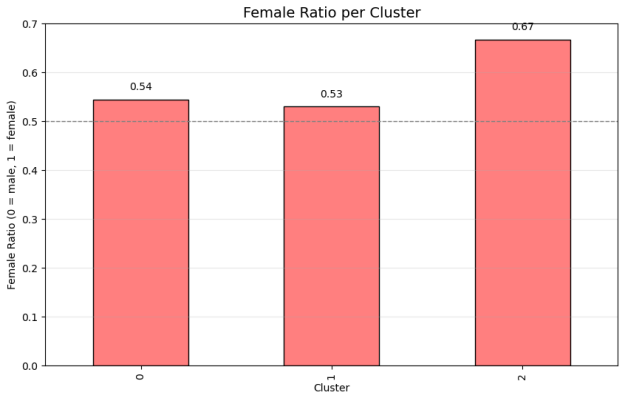


Figure 16: Female ratio in three KMeans clusters (AI-generated pipeline).

Interpretation:

- Cluster 0: older, mid-income, low-spending, low spending intensity (conservative, low-growth segment).
- Cluster 1: young to mid-age, high-income, moderate spending (high potential for premium upselling).
- Cluster 2: very young, low-income but high spending intensity (small but highly engaged, promotion-sensitive segment).

GMM segments. The AI-generated GMM with 6 components yields the cluster means in Table 11. Cluster radar shapes and female ratio per cluster are visualised in Figures 17 and 18.

Table 11: GMM cluster profile means (AI-generated pipeline; key dimensions)

Cluster	Genre	Age	Income (k$)	Score	Spend/Income	Income_tier
0	0.28	59.26	50.18	38.54	0.78	0.77
1	1.00	27.67	71.22	63.15	0.88	1.15
2	0.59	26.48	24.03	66.21	2.88	0.00
3	0.00	29.97	65.90	53.34	0.84	1.00
4	0.47	38.00	99.58	47.03	0.48	2.00
5	1.00	42.68	50.98	41.80	0.86	0.70

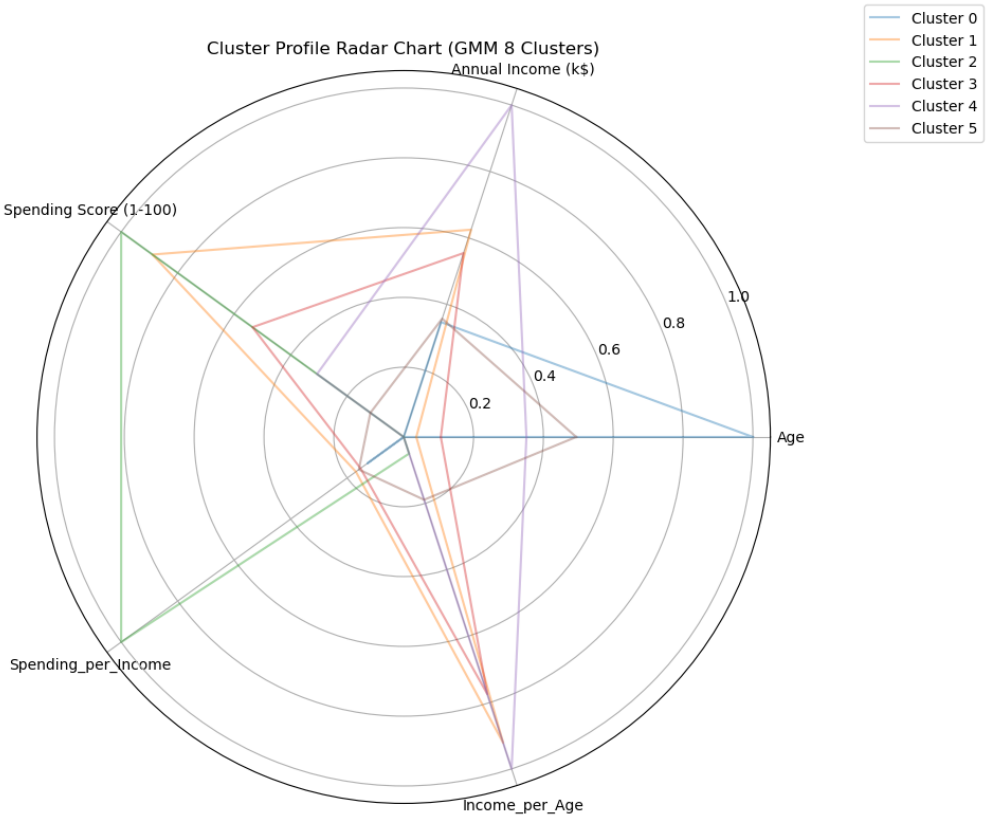


Figure 17: Radar chart for GMM clusters (Age, Income, Score, Spending/Income, Income/Age).

Key insights:

- Gender becomes a strong differentiator: some clusters are all-female (1, 5), some all-male (3), others mixed or majority.
- Income does not fully explain spending: e.g., Cluster 4 has the highest income but moderate score, while Cluster 2 has the lowest income but very high spending intensity.
- High-engagement segments are concentrated among young female clusters (1 and 2) and high-income young males (3); conservative segments are older mid-income groups (0 and 5).

These patterns are consistent with the HDBSCAN-based segmentation and add gender-specific nuance. Combined with the ARI stability results (Table 7), they show that AI-generated pipelines can provide reasonably stable and interpretable baselines, but do not replace careful hyperparameter tuning and domain-informed feature engineering.

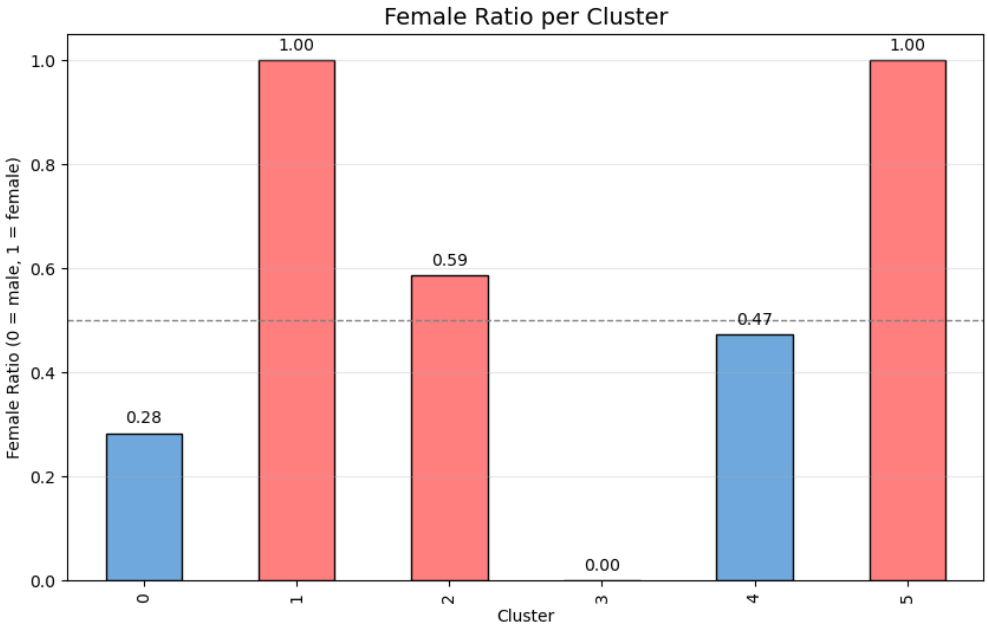


Figure 18: Female ratio by GMM cluster (Genre_encoded; 0 = male, 1 = female).

5 Conclusion

We developed a complete clustering pipeline on the Mall Customer Dataset, covering:

- **Task A:** schema audit, outlier detection using Isolation Forest and LOF, feature engineering (polynomial terms, age and income bins, income-per-age and spending-to-income ratios), and multi-stage feature selection (filter, wrapper, embedded).
- **Task B:** implementation of K-Medoids, GMM and HDBSCAN, systematic hyperparameter search guided by internal indices (Silhouette, CH, DB), low-dimensional visualizations (t-SNE, PCA), and semantic label assignment (Low / Mid-Low / Mid-High / High Income + Outliers).
- **Task C:** LLM-generated KMeans and GMM pipelines with hyperparameter selection via elbow, Silhouette, AIC and BIC, plus stability assessment using ARI.

Among the models considered, HDBSCAN achieves the best internal scores (Silhouette ≈ 0.65, highest CH, lowest DB) and provides explicit noise handling. Its four dense clusters and outlier group can be mapped directly to practical marketing strategies: maintenance for low-income conservative segments, upselling and premium positioning for mid-high and high-income segments, and special handling for atypical high-intensity spenders.

The AI-generated KMeans and GMM pipelines, while weaker in internal metrics, recover a qualitatively similar segmentation structure and highlight additional dimensions such as gender-composition differences across clusters. Their moderate ARI stability shows that LLM-assisted modelling can be a useful complementary tool, but does not obviate the need for principled model selection and visual diagnosis.

6 Limitations and Possible Improvements

6.1 Data and evaluation limitations

The Mall Customer Dataset has several limitations:

- It is small (200 customers) and collected from a single mall, without transaction history, product categories or temporal dynamics. This restricts generalisability and prevents lifecycle or time-series modelling.
- We only rely on internal validity indices (Silhouette, CH, DB, ARI). No external ground-truth labels or business KPIs (e.g., revenue uplift, churn, campaign response) are available to evaluate the economic value of segments.
- Many engineered features are tailored to this specific dataset; they may be mildly overfit and should be re-validated before deployment in other contexts.

6.2 Methodological limitations

On the methodological side:

- HDBSCAN is sensitive to scaling, metric choice and hyperparameters; cluster count and noise ratio can change under alternative settings. A more thorough robustness study (e.g., subsampling, bootstrap) would be useful.
- t-SNE is purely a visualization tool and may distort distances; 2D views should be interpreted qualitatively rather than as precise geometry.
- We restrict attention to classical clustering methods. Deep clustering, representation learning (e.g., autoencoders, contrastive learning) and semi-supervised approaches (using weak business labels) are not explored.

6.3 AI tools and future work

Regarding AI-related tools:

- The LLM-generated code is only tested on a clean, small dataset. Its robustness on large, noisy, high-dimensional or strongly categorical datasets is unclear.
- Future work could:
  - collect richer behavioural and temporal features (visit frequency, basket size, product mix);
  - integrate external business KPIs to judge segments by actual uplift and profitability;
  - conduct cross-dataset stress tests and domain adaptation experiments;
  - adopt deep representation learning to build more expressive embeddings prior to clustering;
  - use LLMs not only for code generation but also for automated documentation, experiment planning and human-readable explanations of segment characteristics.

Overall, the project demonstrates that even with a simple public dataset, careful feature engineering, principled clustering and structured visual analysis can deliver interpretable and action-oriented customer segments, while also illustrating the benefits and limitations of AI-assisted modelling in data mining workflows.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[3] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[4] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[5] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[6] SHWETABH123. Mall customers dataset. `https://www.kaggle.com/datasets/shwetabh123/mall-customers`, 2025. Accessed: 2025-11-27.

A Appendix: Methodology Descriptions

A.1 Model Selection Criteria: AIC and BIC

For Gaussian Mixture Models, we use AIC and BIC to compare candidate models with different numbers of components and covariance structures.

Akaike Information Criterion (AIC). Proposed by Akaike [1]. Let $\hat{L}$ denote the maximised likelihood and k the number of free parameters. The AIC is defined as

$$\mathrm{AIC} = 2k - 2\ln \hat{L}.$$

It penalises model complexity linearly in k.

Bayesian Information Criterion (BIC). Proposed by Schwarz [5]. For sample size n,

$$\mathrm{BIC} = k\ln n - 2\ln \hat{L}.$$

BIC uses a stronger penalty ($\ln n$) and tends to favour more parsimonious models, especially for large n. For both AIC and BIC, lower values indicate a better trade-off between fit and complexity.

A.2 Adjusted Rand Index (ARI)

The Adjusted Rand Index [4] measures the agreement between two partitions of the same dataset, adjusted for chance.

Let n_{ij} be the contingency table between partitions U and V, with row sums $a_i = \sum_j n_{ij}$, column sums $b_j = \sum_i n_{ij}$, and $n = \sum_{ij} n_{ij}$. Using $\binom{x}{2} = x(x-1)/2$, the ARI is

$$\mathrm{ARI} = \frac{\displaystyle\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\Big/\binom{n}{2}}{\displaystyle\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\Big/\binom{n}{2}}.$$

It satisfies:

- ARI $= 1$ if the partitions are identical;
- ARI ≈ 0 for random partitions;
- ARI < 0 if the agreement is worse than random.

A.3 Calinski–Harabasz (CH) Index

The Calinski–Harabasz index [2] quantifies the ratio of between-cluster to within-cluster dispersion.

Let n be the total number of data points, k the number of clusters, μ the global mean, and μ_r the centroid of cluster C_r of size n_r. Define

$$W_k = \sum_{r=1}^{k}\sum_{x\in C_r} \|x-\mu_r\|^2, \qquad B_k = \sum_{r=1}^{k} n_r\|\mu_r-\mu\|^2.$$

The CH index is

$$\mathrm{CH}(k) = \frac{B_k/(k-1)}{W_k/(n-k)}.$$

Higher CH indicates better-defined clusters (higher between-cluster separation relative to within-cluster variance).

A.4 Davies–Bouldin (DB) Index

The Davies–Bouldin index [3] measures average cluster similarity.

For each cluster i, define

$$s_i = \frac{1}{n_i} \sum_{x \in C_i} \|x - \mu_i\|$$

as the average within-cluster scatter, and $d_{ij} = \|\mu_i - \mu_j\|$ as the distance between centroids. For $i \neq j$, define

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}, \qquad R_i = \max_{j \neq i} R_{ij}.$$

The DB index is

$$\mathrm{DB} = \frac{1}{k} \sum_{i=1}^{k} R_i.$$

Lower DB indicates better clustering (more compact and well-separated clusters).

A.5 HDBSCAN Intuition

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) extends DBSCAN by building a density-based cluster hierarchy and extracting stable clusters.

- A mutual reachability distance is defined based on core distances and pairwise distances.
- A minimum spanning tree (MST) of the mutual reachability graph is constructed and used to build a cluster hierarchy as edges are removed in order of distance.
- The hierarchy is condensed using the `min_cluster_size` parameter, pruning branches that never reach this size.
- Cluster stability is computed as persistence over a range of density thresholds, and a non-overlapping set of clusters is chosen to maximise total stability. Unassigned points are labelled as noise (-1).

In our project, the best HDBSCAN configuration uses `min_cluster_size` = 15, `min_samples` = 15, $\varepsilon = 2.0$ and Manhattan metric, yielding 4 dense clusters plus about 12% noise and the best internal metrics among all models.

A.6 Local Outlier Factor (LOF)

LOF is a density-based outlier detector that compares local density of each point to that of its neighbours.

- For each point, compute the k-distance (distance to its k-th nearest neighbour).

- Define the reachability distance between two points as the maximum of the neighbour's k-distance and their actual distance.
- Local reachability density (LRD) is the inverse of the average reachability distance to neighbours.
- LOF is the ratio of the average LRD of neighbours to the LRD of the point.

Points with LOF significantly greater than 1 are considered outliers (locally sparse compared to their neighbours).

A.7 Isolation Forest

Isolation Forest isolates anomalies via random partitioning:

- It builds an ensemble of random trees by recursively splitting data using randomly chosen features and split values.
- Anomalies, being few and different, tend to be isolated in fewer splits (shorter path length from root).
- Normal points require more splits (longer path lengths).
- The anomaly score is derived from the average path length over trees and normalised using the expected path length in random trees.

High anomaly scores indicate points that are easier to isolate and thus more likely to be anomalies.