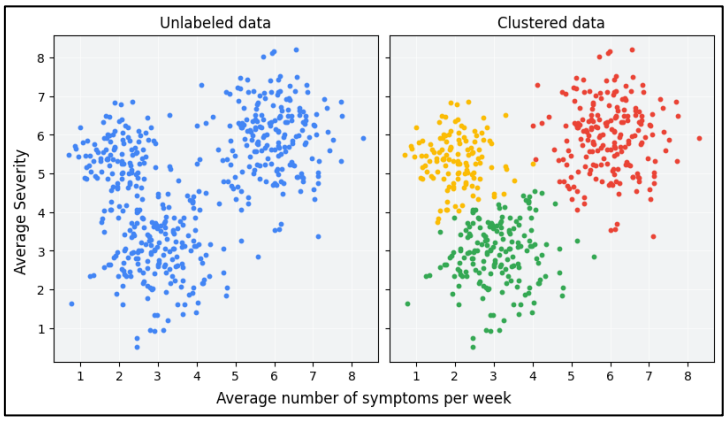
DMBI Experiment 6 – Clustering Algorithms using Rapid Miner Tool

Name: Atharva Lotankar
Class: D15C; **Rnum:** 27
Batch B

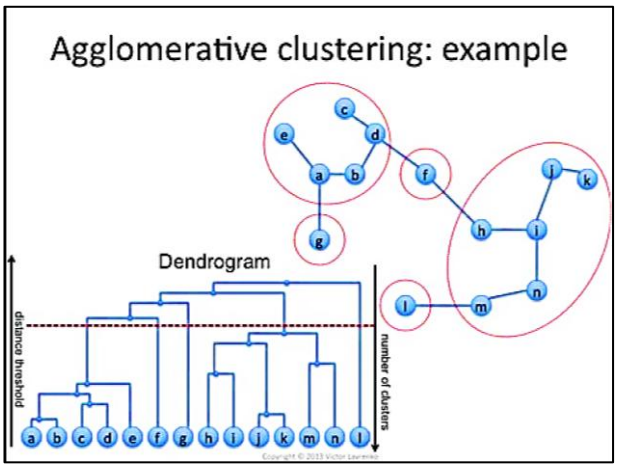
Aim: To implement Clustering Algorithms (k-means, Agglomerative, DBSCAN) using Rapid Miner Tool.

Theory:

Clustering in data mining is an unsupervised learning technique *used to group similar data points without predefined labels*. It helps in pattern discovery, customer segmentation, anomaly detection, and information retrieval.



Among popular methods, **k-means** partitions data into k clusters by minimizing distance to centroids, making it efficient but sensitive to the choice of k and outliers. **Agglomerative clustering**, a hierarchical approach, builds a tree-like structure by successively merging closer clusters, offering interpretability through dendrograms but being computationally expensive for large datasets. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) groups densely packed points and marks low-density regions as noise, making it robust to outliers and suitable for arbitrarily shaped clusters, though sensitive to parameter selection.



RapidMiner, a user-friendly data science platform, supports these clustering techniques with drag-and-drop operators. It allows data preprocessing, visualization, and validation of clusters, making it an effective tool for both beginners and professionals in applying clustering.

Rapid Miner Design Implementation:

K-Means Clustering

Design & Implementation Method

1. **Read Data**: Used the **Read CSV** operator to import the "cosmetics_fraud_2025.csv" dataset into the process.
2. **Handle Missing Values**
    - Added the **Replace Missing Values** operator.
    - Configured it to fill missing data in numeric attributes using a statistical method (e.g., mean or median).
3. **Normalize Features**
    - Inserted the **Normalize** operator to scale all numeric attributes.
    - Ensured that features like CustomerAge, PurchaseAmount, and FootfallCount were on a comparable scale for clustering.
4. **Select Relevant Attributes**
    - Used the **Select Attributes** operator to keep only numeric columns relevant for analysis.
    - Excluded non-numeric fields such as Transaction_ID, focusing on CustomerAge, PurchaseAmount, and FootfallCount.
5. **Apply k-Means Clustering**
    - Added the **Clustering** operator with the algorithm set to k-means.
    - Chose the desired number of clusters (e.g., 5).
    - Connected the processed numeric data to this operator.
6. **Visualize and Interpret Results**
    - Utilized the **Cluster Distance Performance** to create scatter plots, dataset implementation after k-means clusteirng showing clusters with different colours for visual interpretation.
    - Analyzed the data table to verify cluster assignments for each row.
7. **Evaluate Cluster Quality**
    - Checked cluster quality metrics using the Cluster Model evaluation (Davies-Bouldin index, average centroid distances).
    - Reviewed cluster counts and distribution per cluster

Agglomerative (or Hierarchical) Clustering

1. **Replace the Clustering Operator:**
    - Remove the k-means operator from your process.
    - Drag and drop the **Agglomerative Clustering** operator in its place.
2. **Configure the Agglomerative Operator Parameters:**
    - Set the **number of clusters** you want to form, similar to the k-means cluster count.
    - Choose the **linkage method** (e.g., average, complete, single). "Average" linkage is a good default for beginners.
    - Set other parameters if needed (distance metric, etc.).

3. **Retain Data Preparation Stages:**
    - Keep the same **Replace Missing Values**, **Normalize**, and **Select Attributes** operators as before since data processing remains the same.
    - Make sure only numeric columns are selected.
4. **Connect Outputs Correctly:** Connect the output of the "Select Attributes" operator into the input of the Agglomerative Clustering operator.
5. **Run and Visualize:**
    - Execute the process.
    - Use the visualizer to inspect dendrograms (if supported), cluster assignments, and cluster profiles.
6. **Evaluate Clusters:**
    - Check cluster sizes and metrics similarly to before.
    - Agglomerative results are hierarchical, so you can explore clusters at different levels by cutting the dendrogram.

DBSCAN Clustering

1. **Remove Agglomerative Clustering:** Delete the existing Agglomerative Clustering operator from the workflow.
2. **Add DBSCAN Operator:**
    - Insert the **DBSCAN** operator in place of the deleted clustering operator.
    - Connect the input from your Select Attributes operator to the DBSCAN operator.
3. **Set DBSCAN Parameters:**
    - Set **epsilon (eps)**: the maximum distance between two points to be considered neighbors.
    - Set **minPoints**: the minimum number of points required to form a dense region (cluster).
    - These parameters usually need tuning based on data density.
4. **Connect Outputs:**
    - Connect the output of DBSCAN to the **Cluster Model Visualizer** or results output to view clusters.
    - DBSCAN labels some points as noise (outliers), so expect some points not assigned to any cluster.
5. **Run The Process and Evaluate:**
    - Run the workflow.
    - Analyze clusters visually.
    - Identify noise points and cluster groupings that may not be spherical unlike k-means.
6. **Adjust Parameters:** If clusters are too large, too few, or too many noise points, adjust epsilon and minPoints accordingly.
7. **Keep Data Prep Steps:** Retain Replace Missing Values, Normalize, and Select Attributes steps from before.

Inference:

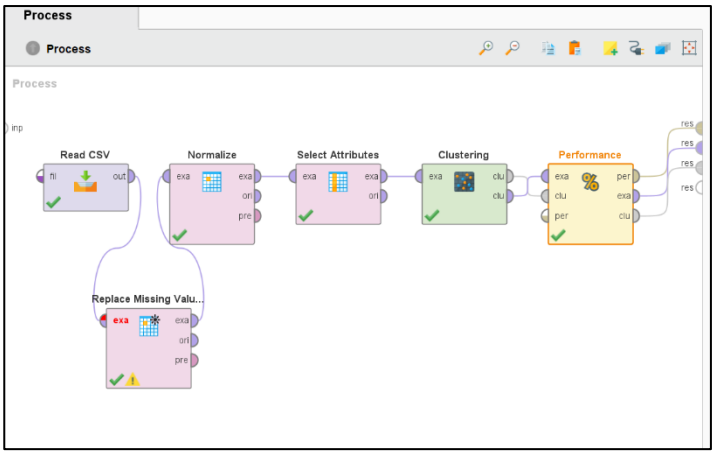
1. **k-means Clustering:**
    - Effective for well-separated, spherical clusters.
    - Requires specifying the number of clusters upfront.
    - Sensitive to outliers and initial centroid placement.

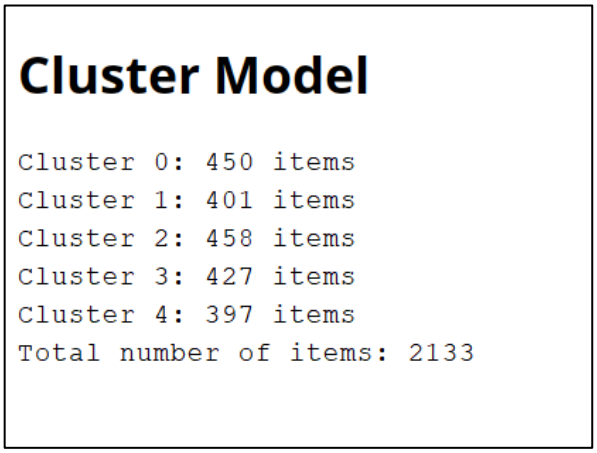
- Provides easily interpretable cluster centroids.
- Visual results show distinct groupings by customer age and purchase amount.

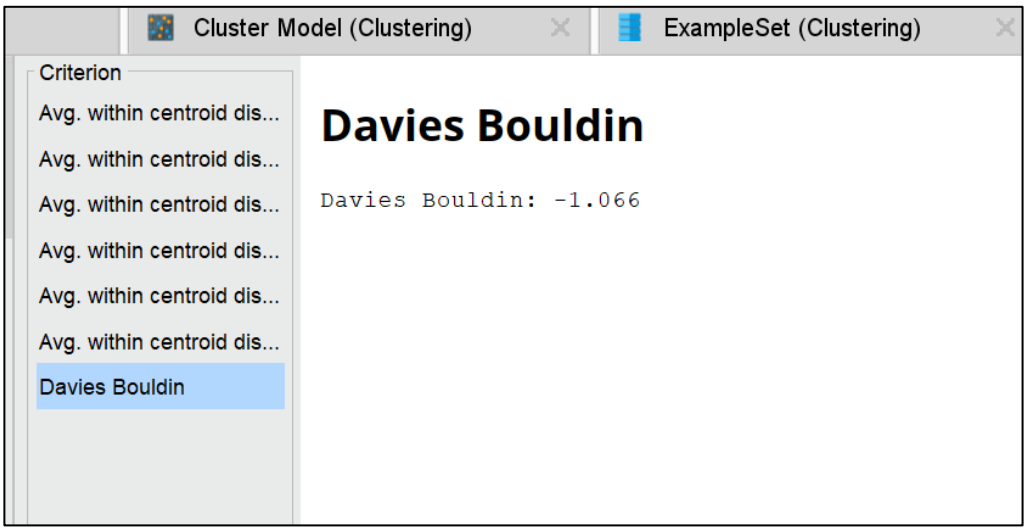
2. **Agglomerative Clustering:**
    - Builds a hierarchy of clusters without predefining the number of clusters.
    - Offers dendrogram visualization for flexible cluster selection.
    - Computation can be slower on large datasets.
    - Suitable for discovering nested cluster structures in data.
3. **DBSCAN:**
    - Detects clusters of arbitrary shape and identifies noise/outliers.
    - Does not require specifying cluster count upfront, but needs tuning of density parameters (eps, minPoints).
    - Handles noisy data better than k-means or hierarchical clustering.
    - Useful for complex datasets with varied density patterns.

Results:

K-Means



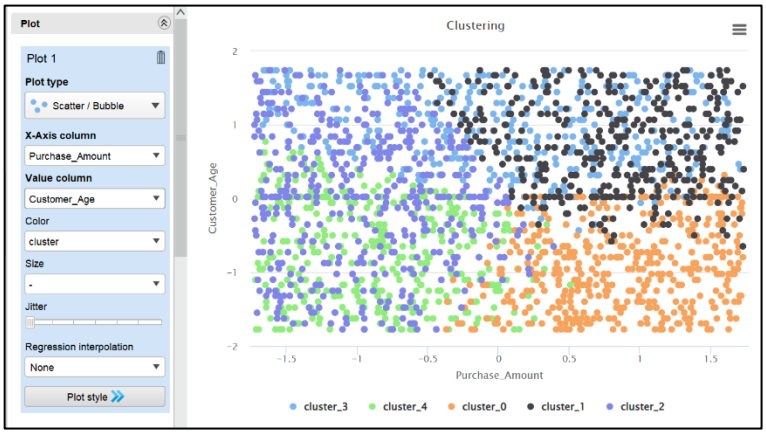




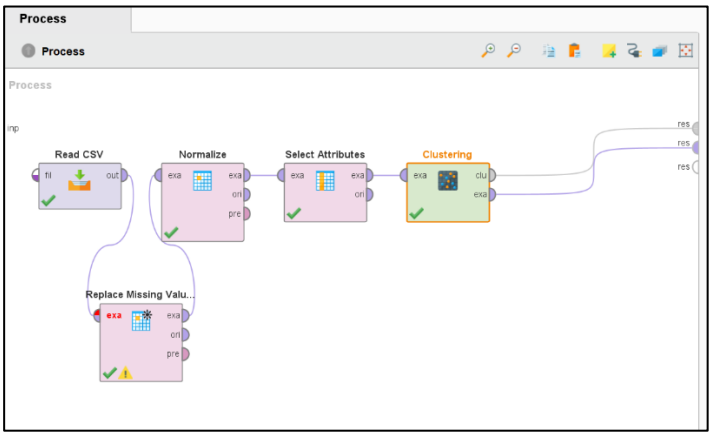
Open in Turbo Prep | Auto Model | Interactive Analysis | Filter (2,133 / 2,133 e

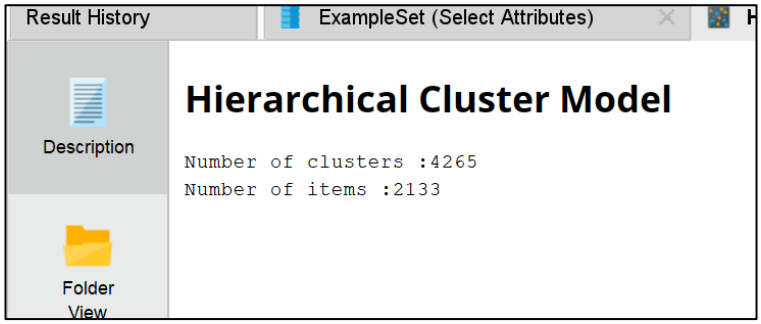
Row No.	id	cluster	Customer_...	Purchase_...	Footfall_Co...
1	1	cluster_3	1.069	-0.227	0.462
2	2	cluster_4	0.322	-1.226	1.018
3	3	cluster_0	-0.725	1.122	-1.346
4	4	cluster_1	1.368	1.497	-0.659
5	5	cluster_1	0.022	0.432	-0.713
6	6	cluster_2	-0.277	-0.536	-0.217
7	7	cluster_2	1.069	-1.192	-0.812
8	8	cluster_4	-0.426	-0.807	1.072
9	9	cluster_4	-0.127	-0.841	1.591
10	10	cluster_0	-1.024	0.398	-1.178
11	11	cluster_0	-1.024	1.545	-1.491
12	12	cluster_4	-0.052	-1.558	0.851
13	13	cluster_2	0.845	-1.721	-1.598

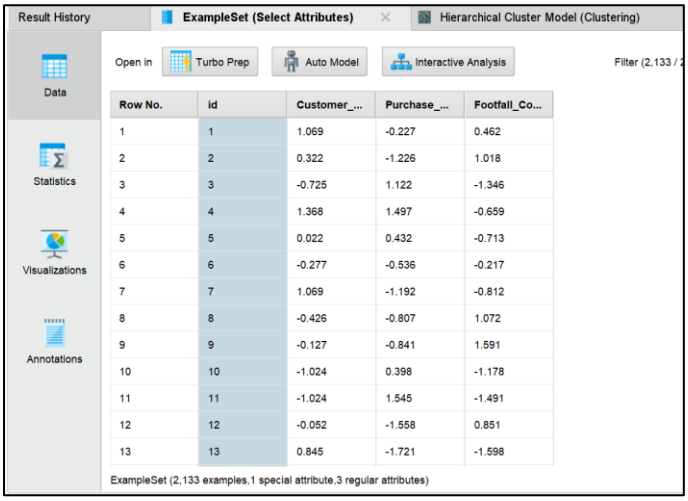
ExampleSet (2,133 examples,2 special attributes,3 regular attributes)



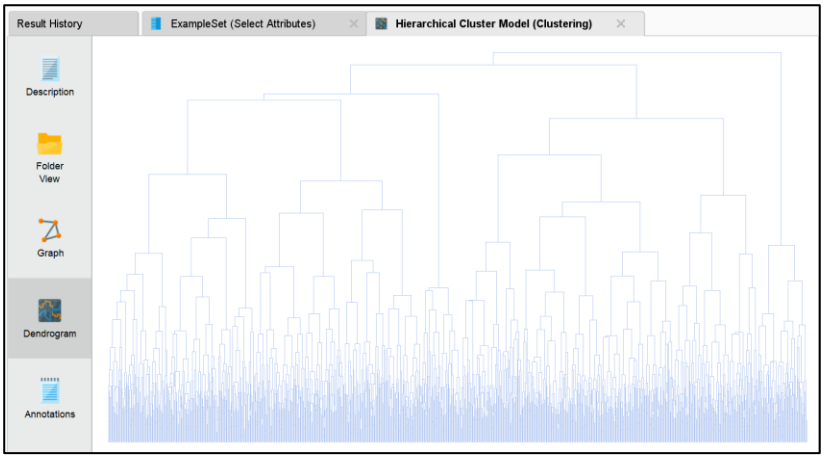
Agglomerative



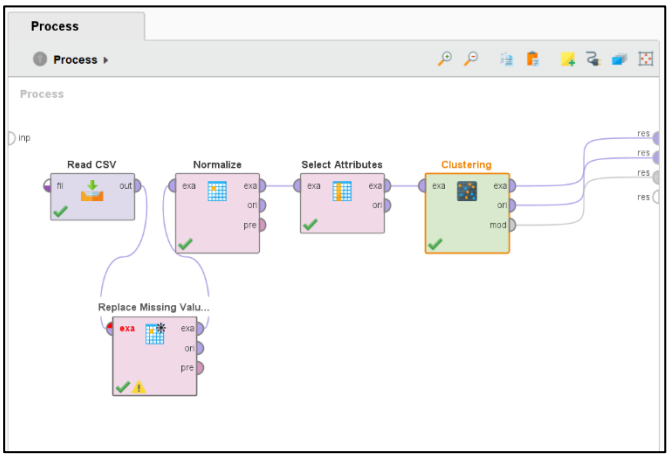


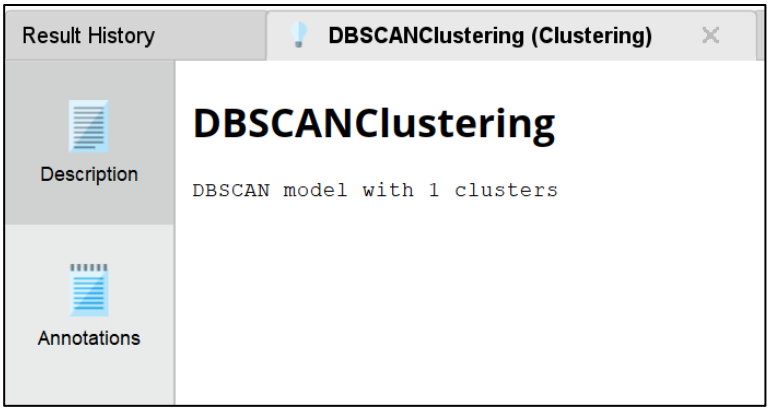


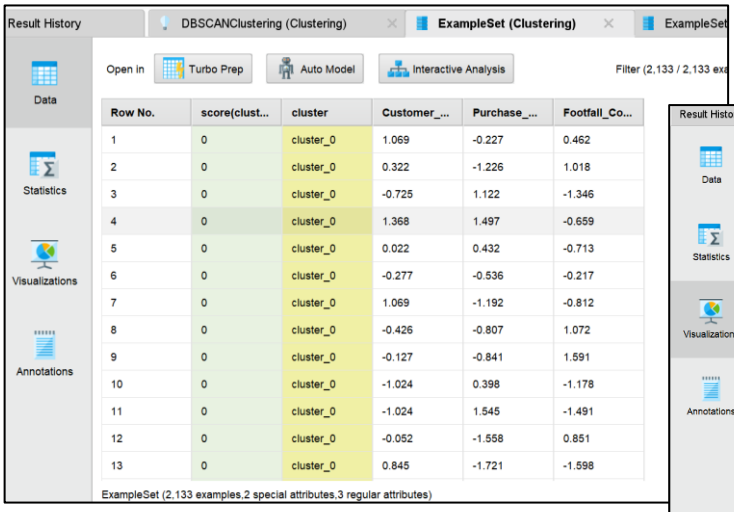
Row No.	id	Customer_...	Purchase_...	Footfall_Co...
1	1	1.069	-0.227	0.462
2	2	0.322	-1.226	1.018
3	3	-0.725	1.122	-1.346
4	4	1.368	1.497	-0.659
5	5	0.022	0.432	-0.713
6	6	-0.277	-0.536	-0.217
7	7	1.069	-1.192	-0.812
8	8	-0.426	-0.807	1.072
9	9	-0.127	-0.841	1.591
10	10	-1.024	0.398	-1.178
11	11	-1.024	1.545	-1.491
12	12	-0.052	-1.558	0.851
13	13	0.845	-1.721	-1.598

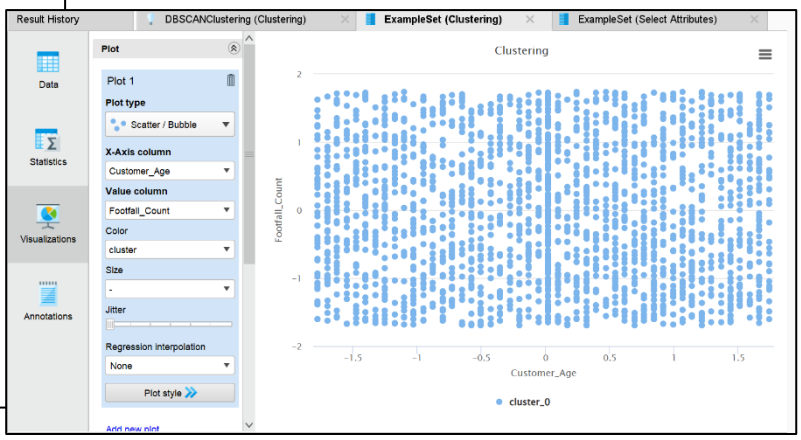


DBSCAN









Conclusion:

The overall clustering experiment using k-means, agglomerative, and DBSCAN algorithms in RapidMiner demonstrated varied capabilities: k-means efficiently partitioned data into spherical clusters but required preset cluster numbers; agglomerative clustering provided hierarchical insights without needing cluster counts upfront, suitable for nested structures; DBSCAN effectively identified clusters of arbitrary shape and noise without specifying cluster numbers, proving robust for complex data distributions. Each method's strengths suggest their contextual suitability, with DBSCAN often outperforming in noisy, irregular data, while k-means and agglomerative offer clear, interpretable cluster formations for well-defined groups. This comprehensive experiment highlights the importance of algorithm choice based on dataset characteristics for meaningful clustering outcomes.