

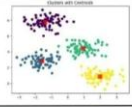
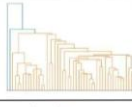
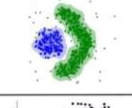
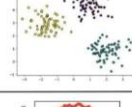
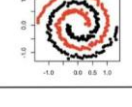
# DMBI Experiment 7 - Clustering Algorithms using Python

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

**Aim:** To implement Clustering Algorithms (k-means, Agglomerative, DBSCAN) using Python.

## Theory:

**K-means, Agglomerative, and DBSCAN** are significant clustering algorithms used to group data points based on similarity. K-means is efficient for spherical clusters with a predefined number, Agglomerative offers hierarchical clustering without predefined groups, and DBSCAN can detect arbitrarily shaped clusters and noise.

Representation	Algorithm Name	Hyperparameter
	K-Means Clustering	Partitions data into K clusters by minimizing variance within each cluster.
	Hierarchical Clustering	Builds a hierarchy of clusters by iteratively merging or splitting existing groups.
	DBSCAN	Forms clusters based on density; groups densely packed points and marks outliers.
	Mean Shift	Finds clusters by locating and adapting to the centroids of data points.
	Spectral Clustering	Uses eigenvalues of similarity matrix to reduce dimensions before clustering.

Python is favoured due to its simplicity, extensive libraries like scikit-learn, and strong community support, making clustering implementation and visualization accessible and efficient.

In real-world applications, these algorithms aid in customer segmentation, fraud detection, image analysis, and more, helping organizations discover hidden patterns and make data-driven decisions across diverse domains.

## Algorithm / Code Snippet

```
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from scipy.cluster.hierarchy import dendrogram, linkage


# Load data

file_path = 'cosmetics_fraud_2025.csv'

data = pd.read_csv(file_path)


# Select relevant numeric columns

numeric_cols = ['Customer_Age', 'Purchase_Amount', 'Footfall_Count']

data_numeric = data[numeric_cols]


# Handle missing values

imputer = SimpleImputer(strategy='mean')

data_imputed = pd.DataFrame(imputer.fit_transform(data_numeric), columns=numeric_cols)


# Normalize data

scaler = StandardScaler()

data_scaled = pd.DataFrame(scaler.fit_transform(data_imputed), columns=numeric_cols)


# K-means clustering

kmeans = KMeans(n_clusters=5, random_state=42)

kmeans_labels = kmeans.fit_predict(data_scaled)


# DBSCAN clustering

dbscan = DBSCAN(eps=0.5, min_samples=5)

dbscan_labels = dbscan.fit_predict(data_scaled)


# Agglomerative clustering

agglo = AgglomerativeClustering(n_clusters=5)

agglo_labels = agglo.fit_predict(data_scaled)
```

```

# Add cluster labels to dataframe

results = data_scaled.copy()

results['kmeans_cluster'] = kmeans_labels

results['dbscan_cluster'] = dbscan_labels

results['agglo_cluster'] = agglo_labels


# Plot function to visualize clustering results

def plot_clusters(data, labels, title, ax):

    scatter = ax.scatter(data.iloc[:, 0], data.iloc[:, 1], c=labels, cmap='tab10', alpha=0.6)

    ax.set_xlabel(data.columns[0])

    ax.set_ylabel(data.columns[1])

    ax.set_title(title)

    plt.legend(*scatter.legend_elements(), title="Clusters")


# Visualize clusters for K-means, DBSCAN, Agglomerative

fig, axs = plt.subplots(1, 3, figsize=(18, 5))

plot_clusters(data_scaled, kmeans_labels, 'K-means Clustering', axs[0])

plot_clusters(data_scaled, dbscan_labels, 'DBSCAN Clustering', axs[1])

plot_clusters(data_scaled, agglo_labels, 'Agglomerative Clustering', axs[2])

plt.tight_layout()

plt.show()


# Generate dendrogram for Agglomerative Clustering

linked = linkage(data_scaled, method='ward')

plt.figure(figsize=(15, 8))

dendrogram(linked,

            orientation='top',

            distance_sort='descending',

            show_leaf_counts=False)

plt.title('Agglomerative Clustering Dendrogram')

plt.xlabel('Sample index')

plt.ylabel('Distance')

plt.show()

```

## **Code Implementation:**

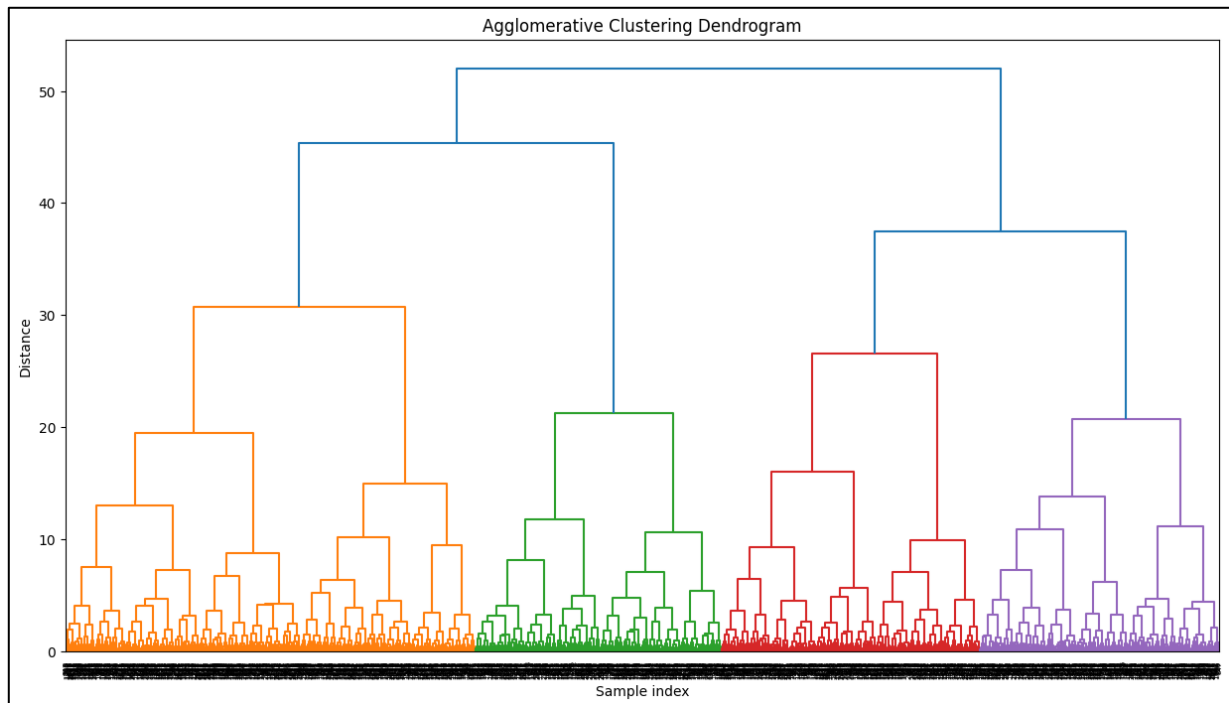
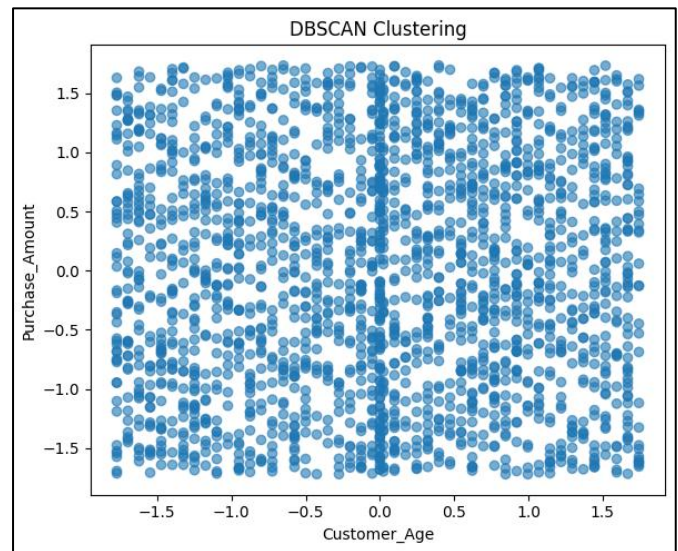
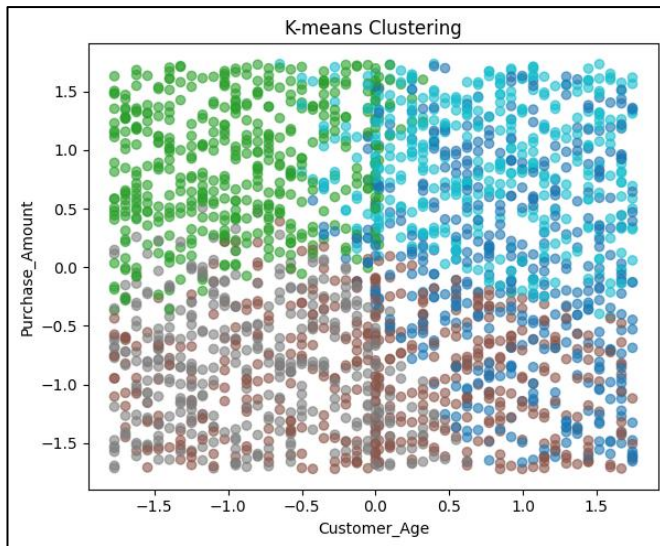
- The experiment begins by importing the dataset and focusing on three numeric variables: Customer\_Age, Purchase\_Amount, and Footfall\_Count, which reflect important features related to customer behavior.
- Missing values in these features are handled by imputing with the mean to ensure complete data for analysis, followed by normalization using standard scaling to place all features on comparable scales, which is crucial for distance-based clustering algorithms.
- Three clustering algorithms are applied: k-means for partitioning data into spherical groups, DBSCAN for identifying clusters of arbitrary shapes and detecting noise, and Agglomerative clustering to reveal hierarchical groupings. Cluster labels are assigned and visualized via scatter plots for side-by-side comparison.
- Additionally, a dendrogram is generated for Agglomerative clustering, allowing visualization of hierarchical relationships between clusters and helping to decide the optimal number of clusters based on linkage distances.

This approach ensures robust preprocessing, diverse clustering perspectives, and comprehensive visualization for insightful customer segmentation analysis.

## **Inference:**

- The Python implementation effectively preprocesses the dataset by handling missing values and normalizing features, which ensures fair input for distance-based clustering methods.
- Applying three distinct clustering algorithms—k-means, Agglomerative, and DBSCAN—provides varied perspectives on the data structure, from partition-based clusters to hierarchical groupings and density-based clusters with noise detection.
- Visualization through scatter plots and dendrograms enhances interpretability, allowing a comparative understanding of cluster shapes, sizes, and hierarchical relationships, thereby supporting informed decision-making in customer segmentation.

## Results:



**Conclusion:** Clustering algorithms implemented in Python, including k-means, Agglomerative, and DBSCAN, effectively segmented customers based on key numeric features. Preprocessing with imputation and normalization ensured reliable results. Visualizations such as scatter plots and dendrograms aided in interpreting cluster structures, supporting data-driven decision making for customer segmentation.