Report on Clustering Results

Several clustering techniques were used in this investigation to group consumer data according to Recency, Frequency, and Monetary (RFM) criteria. The objective was to find trends and combine comparable clients after preprocessing the data and scaling the features.

The quantity of clusters that were formed

- Different hyperparameter selections for DBSCAN resulted in varying numbers of clusters.
- The algorithm first found four clusters with eps=0.5 and min_samples=5. The number of clusters stayed at 4 after adjusting with eps=1.2 and min_samples=3, suggesting that DBSCAN was successful in identifying dense areas in the data.
- With three clusters, other clustering methods like KMeans and Agglomerative Clustering also yielded comparable outcomes.
-

DB Index Value

- One metric used to assess the clusters' cohesion and separation is the Davies-Bouldin (DB) Index. Better clustering performance is indicated by a lower DB Index. The optimal results for DBSCAN were obtained with **eps=1.2 and min_samples=3,** resulting in a **DB Index = 0.32,** indicating a good balance between inter-cluster separation and intra-cluster cohesion.
- The DB Index values for Agglomerative Clustering and KMeans were somewhat higher: Agglomerative Clustering: **0.88;** KMeans: **0.88**
- According to these results, DBSCAN performs better in terms of cluster separation than the other two clustering techniques.
-

Silhouette Score

- An further helpful metric for assessing clustering is the Silhouette Score. A number nearer 1 denotes cohesive and well-separated clusters; the range is -1 to 1.
- The highest Silhouette Score attained by **DBSCAN was 0.51**. This implies that the clustering is quite well-defined, albeit more distinct clusters might result from more optimisation.

Conclusion

With a respectable Silhouette Score and a low Davies-Bouldin Index, DBSCAN outperformed the other algorithms that were tested. eps=1.2 and min_samples=3 were the ideal hyperparameters for DBSCAN, enabling the algorithm to create four clusters while successfully striking a balance between cohesiveness and separation. As indicated by their higher DB Index values, KMeans and Agglomerative Clustering performed satisfactorily but marginally less optimally, especially when it came to cluster separation.