Assignment 5: Report

Sajja Patel - 2021101107, Bhumika Joshi - 2022121006

14 November 2024

1 Part 1

The data was loaded into Power BI. For preprocessing, we went to the transform data option. Here, duplicates and rows with missing values were removed. In addition, we calculated the z-score of the income column and used it to remove all the outliers (z-score beyond +/- 3).

Various visualizations plots have been made. These are:

Bar charts were used to show the distribution of categorical variables.

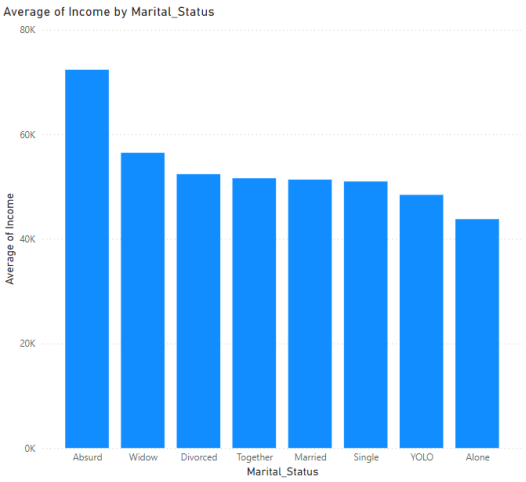


Figure 1: Average Income by Marital Status

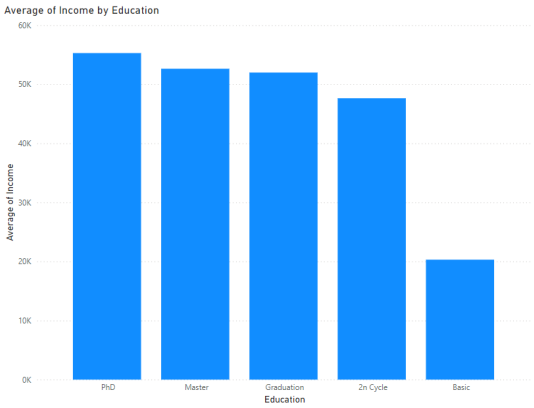


Figure 2: Average Income by Education

We find that people who classify their relationship type as Absurd have the highest average income while those who are Alone have the lowest average income. Further, people with a Phd degree are the richest on average while those with Basic education level are the poorest.

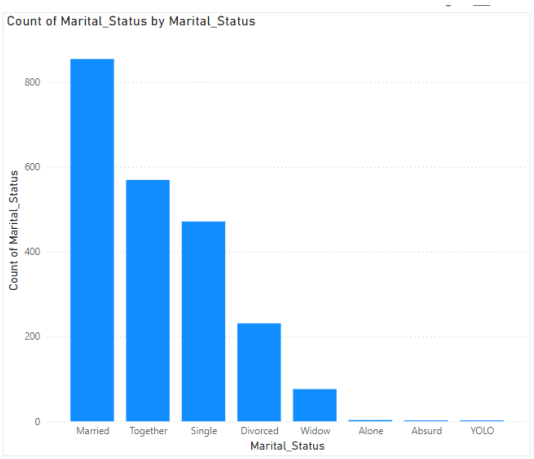


Figure 3: Distribution of Marital Status

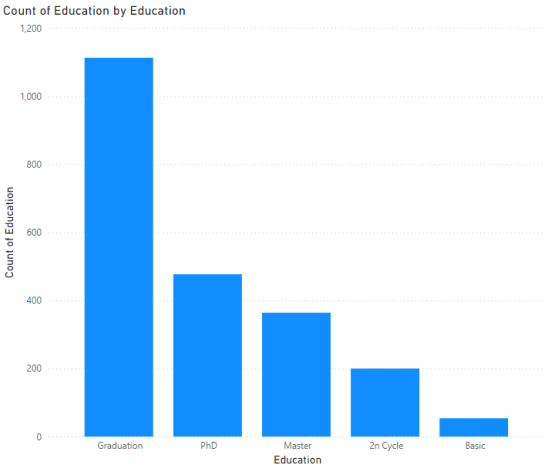


Figure 4: Distribution of Education Level

Continuous numerical values have been shown in the form of histograms.

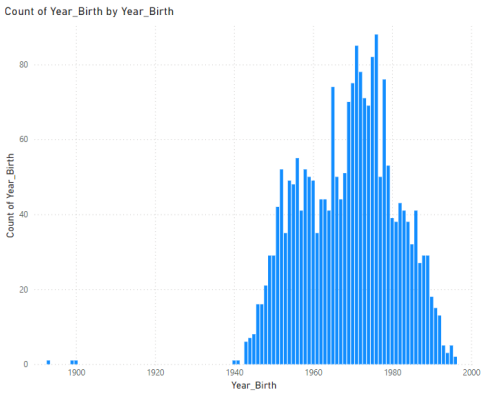


Figure 5: Distribution of Year of Birth

If there are too many unique values, they can be binned as in the Income attribute.

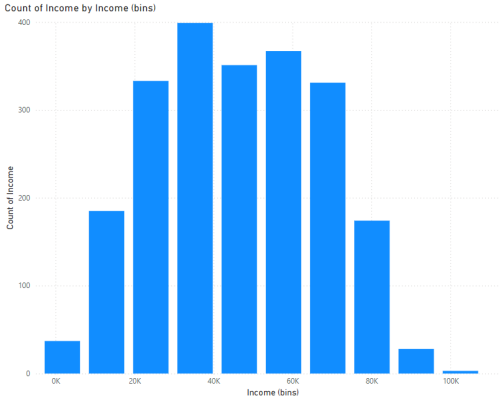
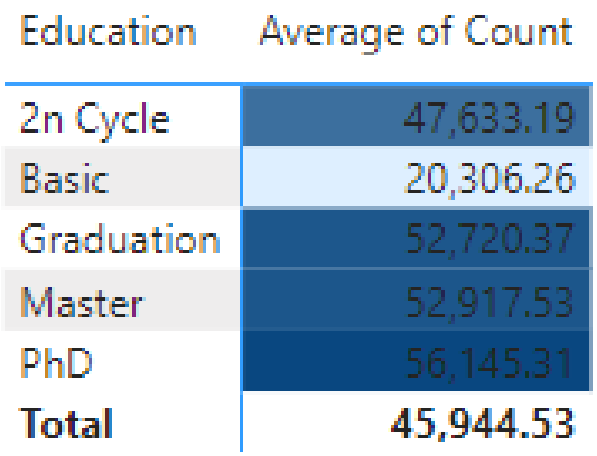


Figure 6: Distribution of Income (Binned)

The data roughly seems to follow a normal distribution.

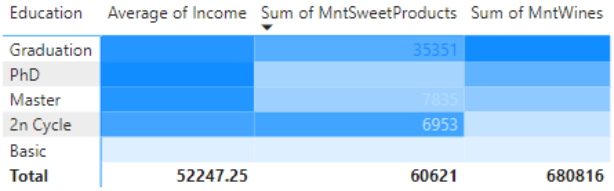
To create heatmaps in Power BI, we loaded the transformed data into the platform and selected the **Matrix** visualization option. We used fields such as **Education**, **Income**, and **Expenditure** (broken down into specific categories like amounts spent on various items). We then created two heatmaps using **conditional formatting**:

- The first heatmap visualized **Education vs. Income (average)**, providing insights into the average income levels across different education categories.



Education	Average of Count
2n Cycle	47,633.19
Basic	20,306.26
Graduation	52,720.37
Master	52,917.53
PhD	56,145.31
Total	**45,944.53**

- The second heatmap displayed **Education vs. Income and Expenditure**, focusing on specific spending categories such as **Expenditure on Sweets** and **Expenditure on Wine**. This allowed us to observe patterns in spending habits alongside income and educational levels.



Education	Average of Income	Sum of MntSweetProducts	Sum of MntWines
Graduation		35351	
PhD			
Master		7835	
2n Cycle		6953	
Basic			
Total	**52247.25**	**60621**	**680816**

Both heatmaps were designed using color gradients to represent the magnitude of the data values, making it easier to interpret trends and relationships in the dataset.

Heatmaps can also be designed by running the appropriate python script for the desired correlation heatmap in Power BI.

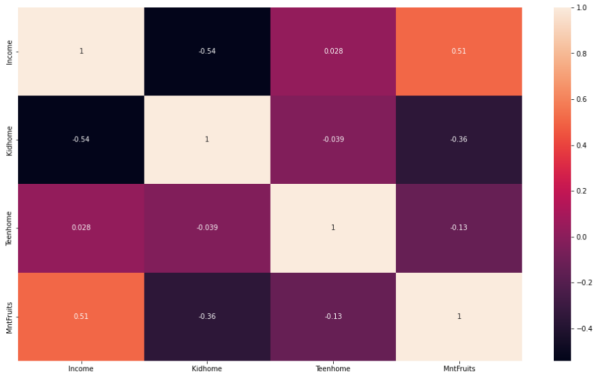


Figure 7: Correlation between Income, Teenhome, Kidhome and MntFruits

Income seems to be highly negatively correlated with Kidhome and positively correlated with MntFruits.

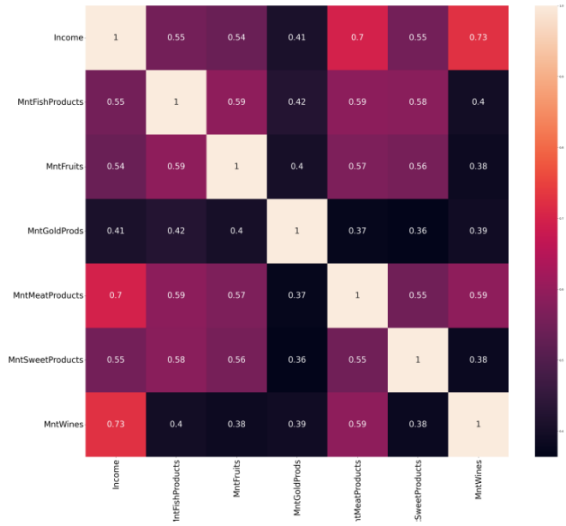


Figure 8: Correlation Between the Purchase of Different Items

We see here that higher income correlates with Higher Meat and Wine purchases and purchases are all positively correlated with income.

2 Part2

The K-Means clustering implementation here represents a fundamental unsupervised learning algorithm designed to partition data points into k distinct clusters. This approach begins with a user-defined number of clusters (k) and iteratively refines the cluster assignments and centroids to minimize intra-cluster variance, often referred to as inertia.

The algorithm has been implemented from scratch to provide a deeper understanding of its mechanics, avoiding reliance on pre-built clustering libraries. The steps include:

- Random Initialization: The centroids are initialized by selecting kk random points from the dataset.
- Cluster Assignment: Each data point is assigned to the nearest centroid, measured by Euclidean distance.

- Centroid Update: The centroids are recalculated as the mean of the points in each cluster.
- Iteration and Convergence: These steps repeat until the centroids stabilize or a predefined iteration limit is reached.

This process ensures that the algorithm finds locally optimal cluster assignments based on the data structure.

Determining the optimal number of clusters (kk) is critical for effective clustering. This implementation uses:

- Elbow Method: It evaluates the sum of squared distances (inertia) for different kk values. A sharp decline followed by a plateau in inertia indicates the "elbow," suggesting the best trade-off between cluster compactness and overfitting.
- Silhouette Score: This metric measures how similar a point is to its own cluster compared to other clusters. Higher silhouette scores indicate better-defined and more distinct clusters.

The algorithm leverages dimensionality reduction through PCA (Principal Component Analysis) to project the high-dimensional data into 2D (or 3D) space. This allows for intuitive visualization of clusters and their centroids. Such plots help in visually inspecting the separation and compactness of clusters, complementing numerical metrics.

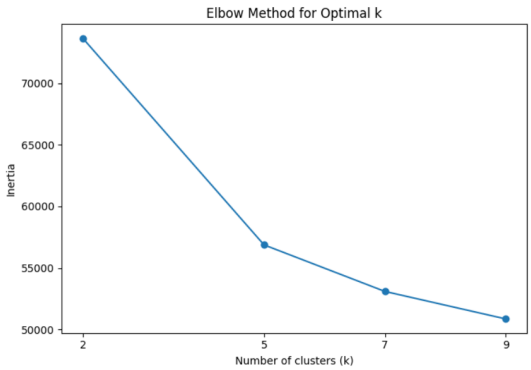


Figure 9: Inertias for Various k Values

Figure 10: Silouhette Scores for Various k Values

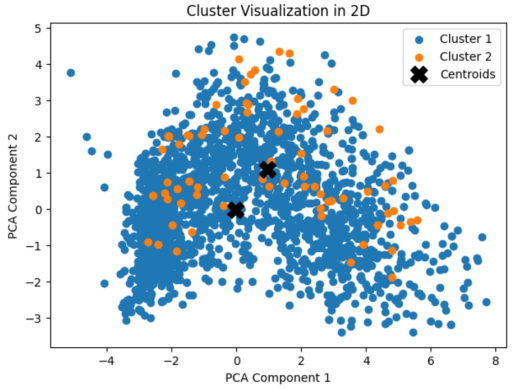


Figure 11: 2 Clusters

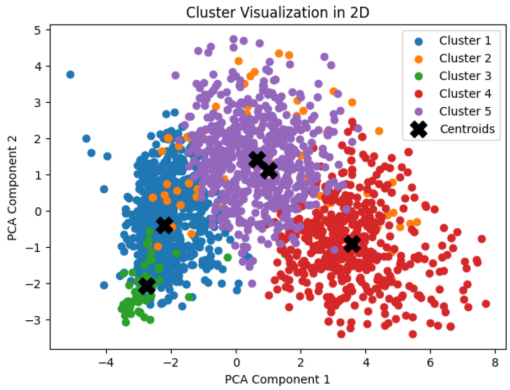


Figure 12: 5 Clusters

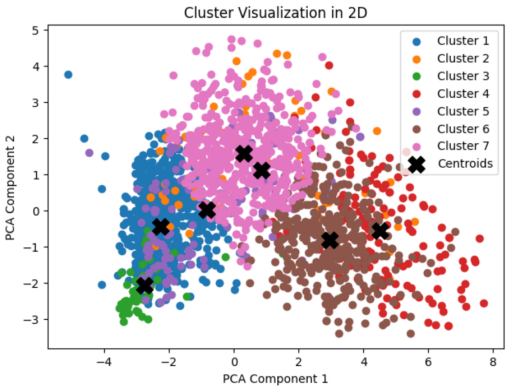


Figure 13: 7 Clusters

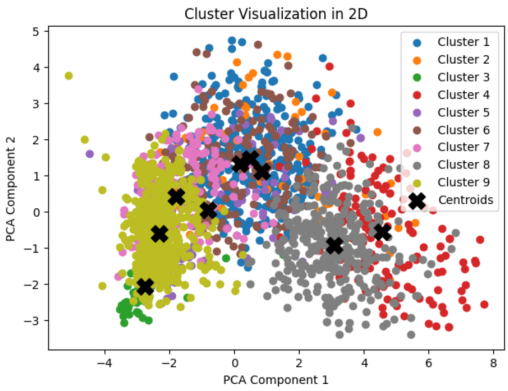


Figure 14: 9 Clusters

According to the elbow method and silhouette scores, 9 seems to be the optimal number of clusters for the given dataset.

3 Part 3

Before applying any of the algorithms ,we have first made a scatter plot of x1 vs x2 for all the datasets.

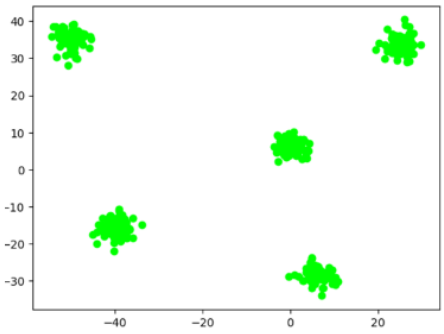


Figure 15: Scatter plot of Compact.csv

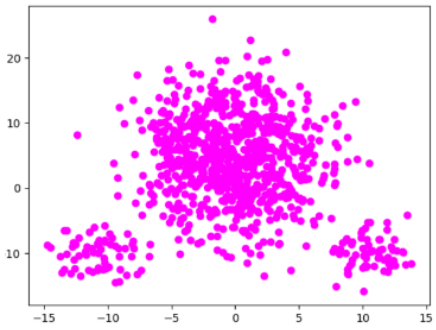


Figure 16: Scatter plot of Skewed.csv

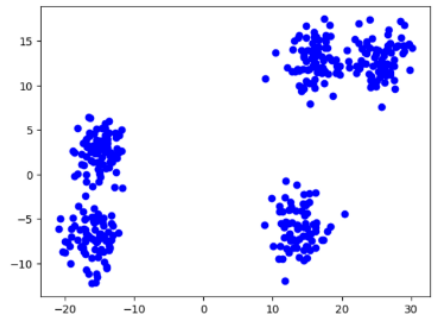


Figure 17: Scatter plot of SubClusters.csv

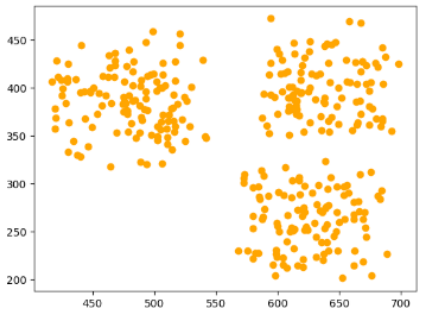
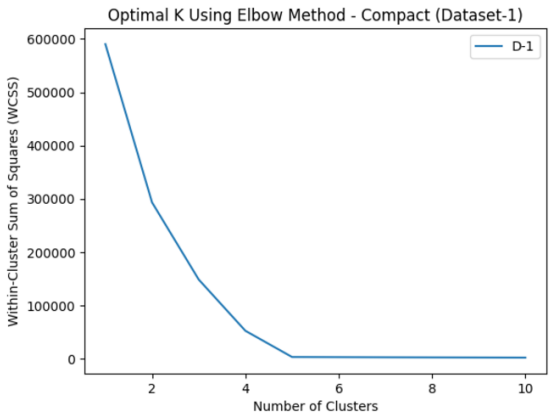
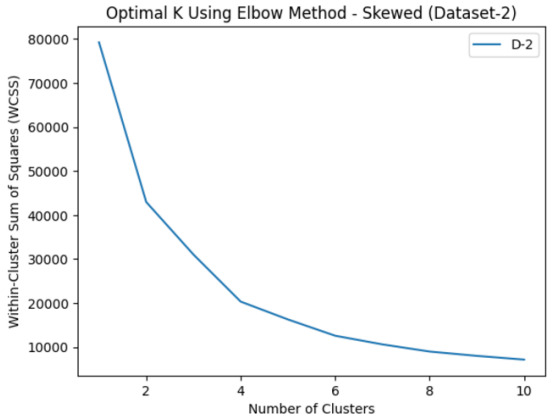


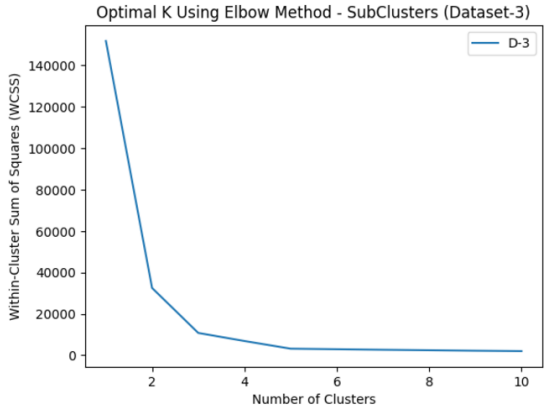
Figure 18: Scatter plot of WellSeparated.csv

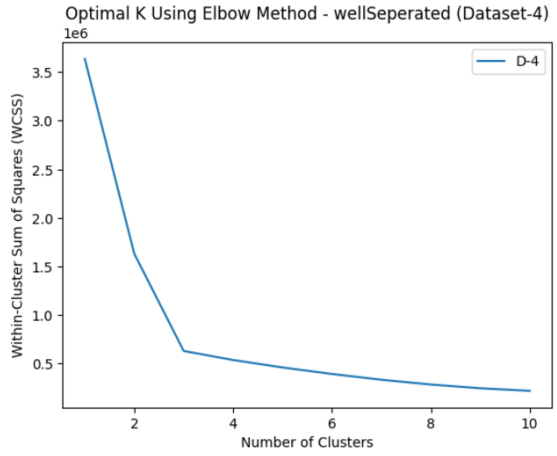
3.1 K-Means Clustering

The code applies K-Means clustering on each of the four datasets, testing different cluster counts (from 1 to 10) and calculating the inertia (within-cluster sum of squares) for each value of 'k'. The K-Means model is initialized using the k-means++ method to accelerate convergence and is run for a maximum of 300 iterations with 10 different initializations to ensure stable results. The inertia values for each cluster count are stored and then plotted using the ***Elbow Method***. This method helps visualize the relationship between the number of clusters (k) and the inertia, with the optimal cluster count indicated by the "elbow" point, where the reduction in inertia starts to slow down. If the dataset contains non-numeric columns, the code prints a message indicating that clustering cannot be performed on that dataset.



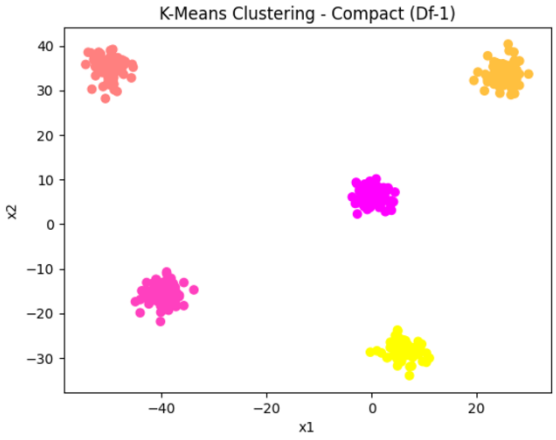


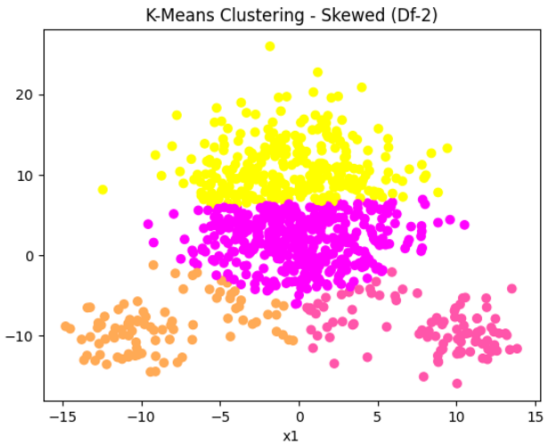


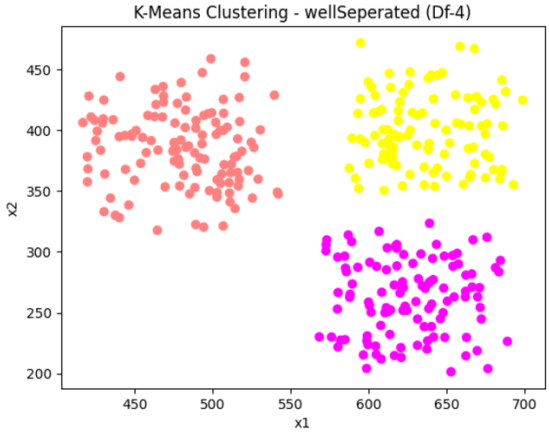


From the above plots, we come to a conclusion of choosing the optimal_cluster_counts as 5, 4, 3, 3 respectively for the 4 datasets.

For each dataset, the code calculates two metrics: silhouette score and inertia. The silhouette score measures how similar the data points within a cluster are, with higher values indicating better clustering. The inertia value represents the sum of squared distances between each sample and its corresponding cluster center, reflecting the compactness of the clusters. The results of K-Means clustering are visualized using scatter plots, with data points colored according to their cluster assignment. After processing all datasets, the code outputs the silhouette scores and inertia scores for each dataset, which can be used to assess the quality of clustering and compare the clustering performance across the datasets.







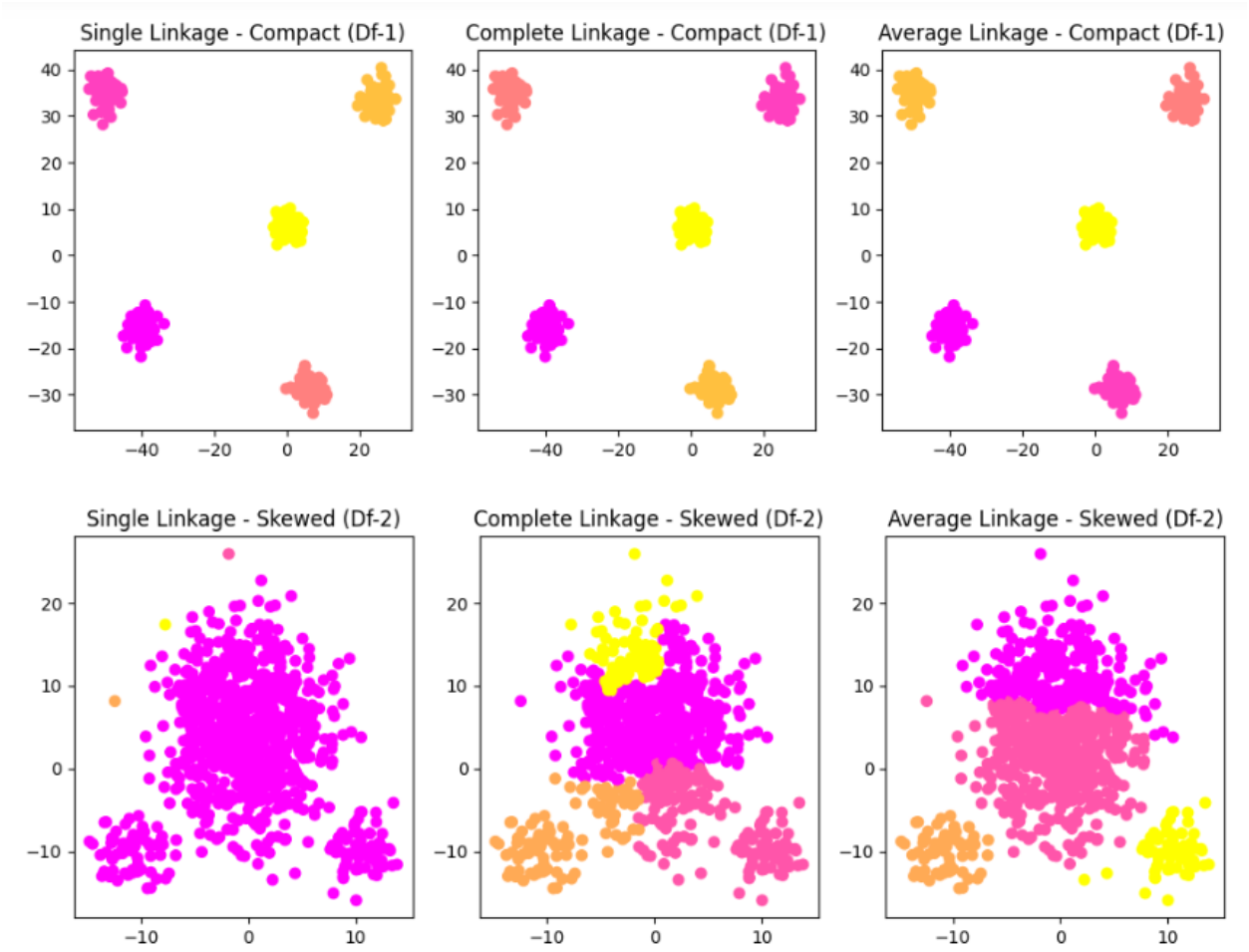
OBSERVATIONS:

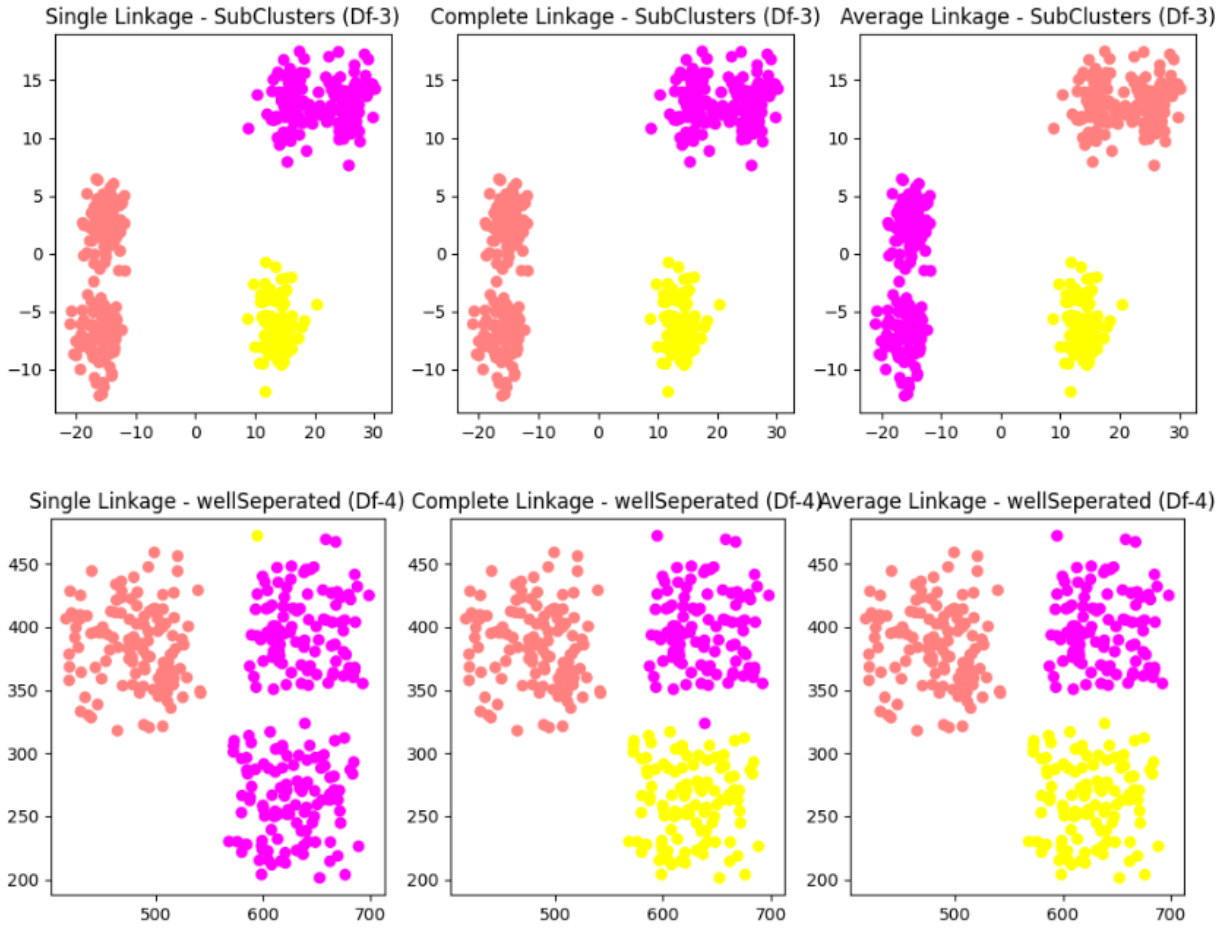
- For the Compact Data, the K-Means algorithm performs well, as the clusters are spherical and of equal size, which matches the algorithm's assumptions.
- In the case of Skewed Data, the algorithm faces challenges because it assumes clusters to be spherical and of similar size, which is not true for this dataset.
- The algorithm performs well with the SubClusters Data, as it correctly assigns points to the nearest centroid, resulting in effective clustering.
- Finally, for Well-Separated Data, K-Means works effectively due to the clear separation and similar size of the clusters, making it easier for the algorithm to identify distinct groups.

3.2 Aggolomerative Clustering

The code applies Agglomerative Clustering to multiple datasets using three different linkage strategies: single linkage, complete linkage, and average linkage. It first iterates through each dataset, setting up the AgglomerativeClustering model for each linkage type with an optimal cluster count. For each model, the silhouette score (a measure of clustering quality) is computed, as well as the inertia, which is the sum of squared distances of samples to the mean of their respective clusters.

The silhouette score is calculated using the silhouette_score function, and inertia is manually calculated by summing the squared differences between the data points and the cluster center for each cluster, ignoring noise points. After fitting the models, the silhouette scores and inertia values for each linkage method are stored in separate lists.





OBSERVATIONS:

- For the Compact Data, Agglomerative Clustering methods like complete or average linkage are particularly effective because these linkage strategies emphasize the maximum or average distance between clusters, respectively, which works well for datasets with compact and spherical clusters.
- For the Skewed Data, Agglomerative Clustering, like K-Means, encounters challenges due to the data's uneven structure. The method struggles to identify meaningful clusters, especially when they are not spherical or uniformly distributed.
- When it comes to the Subcluster Data, the method performs well, particularly when using average or complete linkage. These strategies are better suited to identifying subclusters within the larger clusters, as they take into account the overall distance between clusters, making them more capable of handling such structures.
- For the Well-Separated Data, Agglomerative Clustering works particularly well, especially with single or complete linkage. These methods are ideal for datasets with distinct, well-separated clusters, where the clear boundaries between clusters make the clustering process more effective.

4 DBSCAN Algorithm

This code is focused on applying DBSCAN clustering to `df4` while optimizing its parameters, ε (radius for a neighborhood) and `min_samples` (minimum points to form a core point). It explores a range of ε values (from 1 to 29, spaced into 5 values) and `min_samples` values (from 1 to 50) to find the combination that maximizes the silhouette score, which measures how well-defined the clusters are.

For each pair of parameters, DBSCAN is run, and the number of clusters is calculated (ignoring noise labeled as -1). If the dataset has more than one cluster, the silhouette score is computed. The combination of ε and `min_samples` yielding the highest silhouette score is selected as the best configuration.

After determining the optimal parameters, DBSCAN is applied to `df4` with these values, and the resulting clusters are visualized on a scatter plot, where points are colored based on their cluster labels.

The process was done separately for `df4` because its data distribution is fundamentally different from the other datasets. The earlier datasets have had more complex or skewed distributions with overlapping or unclear boundaries, necessitating a denser search over smaller ε values (e.g., 1–5). In contrast, `df4` has well-separated clusters, so a wider range of ε values (e.g., 1–29) is more appropriate to ensure DBSCAN captures the natural separations efficiently without excessive fine-tuning.

This differentiation in approach ensures that the parameter search aligns with the dataset's characteristics, avoiding unnecessary computation or suboptimal clustering results. The well-separated nature of `df4` makes it easier to optimize clustering with a broader parameter space. ***OBSERVATIONS:***

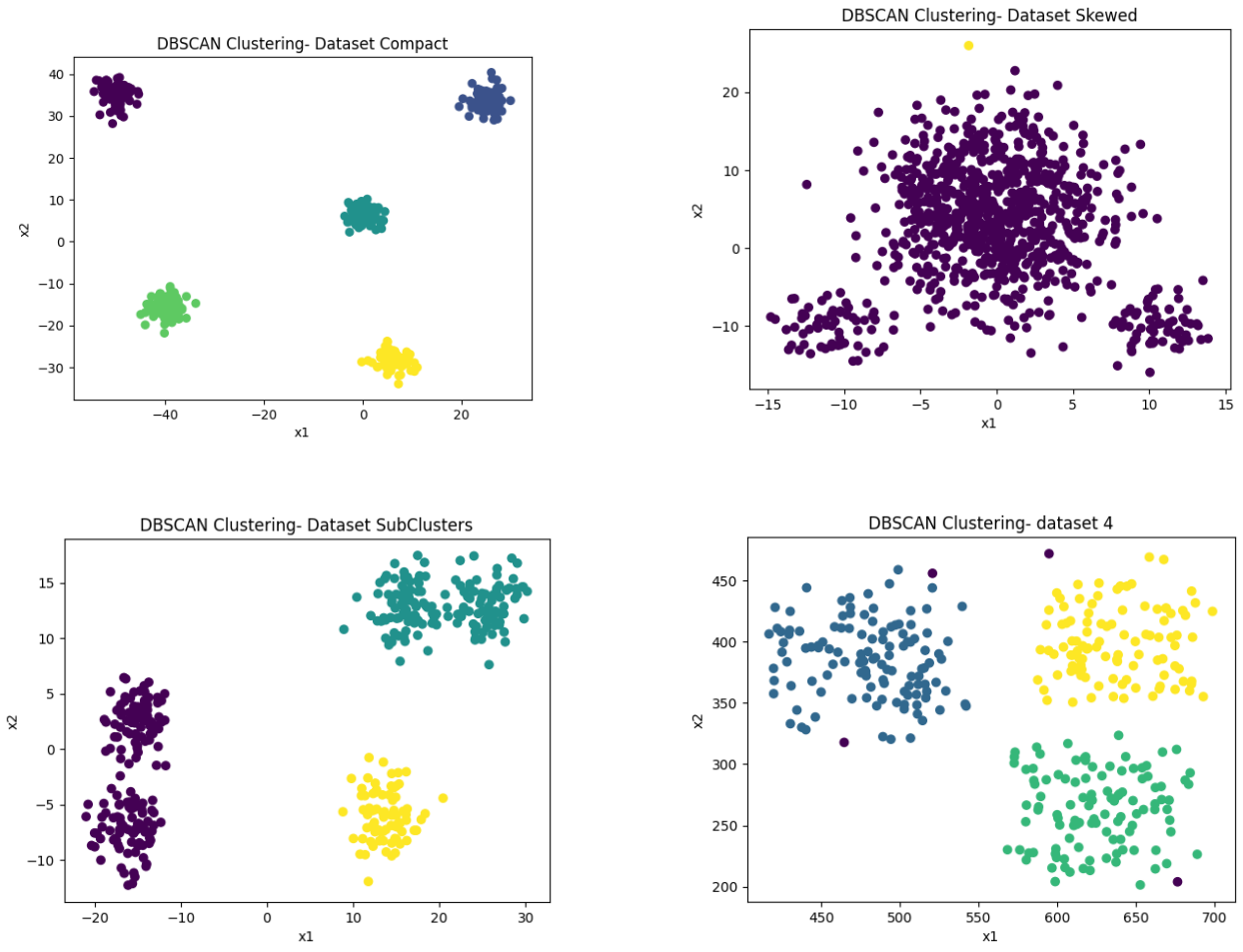


Figure 15: Overall caption for the 2x2 image grid

- Compact Data: DBSCAN performs well here as the clusters have consistent densities and well-defined boundaries, making them easy to detect.
- Skewed Data: Handles skewed clusters effectively since it does not rely on predefined cluster shapes like KMeans and adapts to the data's inherent structure.
- SubClusters Data: Performs well with subclusters due to their sufficient density, but precise parameter tuning is essential to distinguish between closely packed clusters.
- Well-Separated Data: Although not specifically designed for well-separated clusters, it works well in this case due to the high density within clusters, provided the parameters are carefully fine-tuned.

5 Silhoutte Score

The silhouette scores across the datasets and clustering algorithms reveal key insights into their performance. For the **Compact dataset**, all algorithms achieve a perfect or near-perfect silhouette score of 0.91, indicating that compact clusters are well-defined and easily separable, regardless of the method used. For the **Skewed dataset**, DBSCAN outperforms other algorithms with a silhouette score of 0.48, highlighting its ability to handle uneven cluster shapes and densities better than K-Means or Agglomerative clustering. In contrast, the Single linkage method performs poorly (0.17) due to its sensitivity to noise and outliers. The **SubClusters dataset** yields consistent scores (0.73) across all methods, suggesting that the clusters are sufficiently dense and well-separated for all algorithms to identify them effectively. Finally, for the **Well-Separated dataset**, K-Means and Agglomerative (Average and Complete linkage) achieve similar scores (0.61), indicating their capability to separate clusters well. However, DBSCAN slightly underperforms (0.58), as it is not specifically designed for this type of data but still performs reasonably well with proper parameter tuning.

Silhouette scores, ranging from -1 to 1, measure how well each data point fits within its assigned cluster relative to other clusters. A score close to 1 indicates well-separated and cohesive clusters, while a score near 0 suggests overlapping or ambiguous clusters. Negative values imply incorrect clustering. These scores help assess the effectiveness of the clustering algorithms for each dataset, emphasizing the importance of selecting the right method based on the data's characteristics.

	Dataset	K-Means	Agg. Single	Agg. Complete	Agg. Average	DBSCAN
0	Compact	0.911780	0.911780	0.911780	0.911780	0.911780
1	Skewed	0.406510	0.173779	0.292751	0.391412	0.483642
2	SubClusters	0.738949	0.738949	0.738949	0.738949	0.738949
3	wellSeperated	0.613946	0.346804	0.612632	0.613946	0.588947

Figure 16: Silhoutte score for datasets across diff algos

6 Inertia Score

The inertia values across datasets and clustering algorithms provide insights into their performance in minimizing within-cluster variance. For the **Compact dataset**, K-Means achieves the lowest inertia value (3163.69), signifying its ability to form tight, cohesive clusters, while all Agglomerative methods (Single, Complete, Average) report significantly higher inertia (5.90×10^5), indicating less compact clustering. In the **Skewed dataset**, K-Means again performs better ($20,328.52$) with lower inertia compared to Agglomerative methods (7.92×10^4), although the results are closer, reflecting challenges in capturing the irregular cluster shapes. For the **SubClusters dataset**, K-Means reports the smallest inertia ($10,795.40$), showing its effectiveness in separating subclusters, while Agglomerative methods yield consistently higher values (1.52×10^5). Finally, in the **Well-Separated dataset**, K-Means also demonstrates lower inertia (6.29×10^5) compared to Agglomerative clustering, whose inertia values are the highest across all datasets (3.63×10^6).

Interpretation: Inertia measures the sum of squared distances of data points to their respective cluster centroids. Lower inertia signifies tighter clusters, but it does not account for cluster separation or irregular

	K-Means Inertia	Agg. Single Inertia	Agg. Complete Inertia	\
0	3163.693808	5.903962e+05	5.903962e+05	
1	20328.527705	7.924804e+04	7.924804e+04	
2	10795.396187	1.517436e+05	1.517436e+05	
3	629926.614046	3.635559e+06	3.635559e+06	

	Agg. Average Inertia
0	5.903962e+05
1	7.924804e+04
2	1.517436e+05
3	3.635559e+06

Figure 17: Inertia score for diff datasets across algos

shapes. While K-Means excels in datasets with compact or spherical clusters (low inertia), Agglomerative methods are less effective for such configurations, as seen in their higher inertia values. However, inertia should be interpreted in conjunction with other metrics (like silhouette scores) for a more comprehensive evaluation of clustering performance.