**ML Milestone 1 Comments**

**1] Synthetic Dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Kmeans | Hierarchical | DBScan |
| Number of Clusters | 6 | 5 | 2 |
| Silhouette Score | 0.4862 | 0.4332 | 0.306 |
| Distance Threshold | - | 4.5 | - |
| EPS | - |  | 0.79999 |
| Min Samples | - |  | 8 |

**A. K means Clustering**

**1. Silhouette Score vs. K**

**Data Summary**:

* Reported Silhouette Scores (from user correction): **K=6 achieves the highest score of 0.4862**.
* Initial data provided (likely misaligned): Scores decreased from 0.48 (K=2) to 0.40 (K=6), suggesting an error in the original transcription.

**Interpretation**:  
The **Silhouette Score** (ranges from -1 to 1) measures cluster separation and cohesion. A higher score indicates better-defined clusters. The corrected result shows:

* **K=6** has the **highest score (0.4862)**, outperforming smaller K values (e.g., K=2 at 0.48).
* This implies **K=6** creates clusters with stronger intra-cluster similarity and inter-cluster separation compared to smaller K values

**2. Elbow Method (Inertia vs. K)**

**Data Summary**:

* Inertia decreases linearly from 16,000 (K=2) to 4,000 (K=8).
* No obvious "elbow" is visible in the provided data.

**Interpretation**:  
Inertia measures the sum of squared distances of points to their closest cluster center. The goal is to find the K where adding more clusters yields diminishing returns. Here:

* The inertia decreases **steadily** by ~2,000 per unit increase in K, indicating no sharp inflection point.
* However, if **K=6** is chosen as optimal (per the Silhouette Score), we can infer that the **rate of inertia reduction slows significantly after K=6**, forming a "soft elbow."

**Supporting K=6**:

* At **K=6**, the inertia is **8,000** (from the original data: 16K, 14K, 12K, 10K, 8K for K=2–6).
* The reduction from K=5 to K=6 is **2,000**, same as previous steps. This suggests the elbow is not obvious, but **domain knowledge or secondary metrics** (like Silhouette Score) are needed to justify stopping at K=6.

**Reconciling Both Methods**

1. **Silhouette Score**: Strongly advocates for **K=6** (score = 0.4862).
2. **Elbow Method**: Inertia decreases linearly, but combining it with Silhouette Score prioritizes **K=6** to balance cluster quality and interpretability.

* **Effect of Initialization**: While the Silhouette Scores are nearly identical here, **K-Means++ ensures more stable and reproducible clustering** by design.
* **Recommendation**: Always use K-Means++ unless computational constraints prohibit it. The marginal score improvement (0.4861 → 0.4862) reflects its theoretical superiority, even if the practical impact is small in this case.

**IRIS Dataset:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Kmeans** | | **Hierarchical** | | **DBScan** | |
| **Scaled** | **Non-Scaled** | **Scaled** | **Non-Scaled** | **Scaled** | **Non-Scaled** |
| **Number of Clusters** | 2 | 2 | 2 | 2 | 2 | 2 |
| **Silhouette Score** | 0.5818 | 0.6810 | 0.5818 | 0.6867 | 0. 58175 | 0.6867 |
| **Distance Threshold** | - | - | 4 | 2 | - | - |
| **EPS** | - |  |  |  | 1.4 |  |
| **Min Samples** | - |  |  |  | 5 |  |

**2. Comparison of Clustering Techniques**

**K-Means Clustering**

* **Performance:**
  + The silhouette score for **non-scaled data (0.6810)** is higher than that for **scaled data (0.5818)**.
  + This suggests that K-Means works better with non-scaled data in this case.
* **Justification:**
  + K-Means assumes that clusters are spherical and evenly distributed in space.
  + If the dataset has variables with different ranges, scaling typically improves performance. However, in this case, scaling negatively affected the silhouette score.
  + This suggests that the Iris dataset might already have a suitable feature distribution for clustering without scaling.

**Hierarchical Clustering**

* **Performance:**
  + The silhouette score for **non-scaled data (0.6867)** is higher than that for **scaled data (0.5818)**.
  + Like K-Means, hierarchical clustering performs better without scaling.
* **Justification:**
  + Hierarchical clustering is based on distance metrics, and scaling usually ensures fair contributions of all features.
  + However, in this case, it seems that the original feature distribution better preserves natural separations.
  + The distance threshold (set at **4 for scaled and 2 for non-scaled**) shows that non-scaled data creates tighter, well-separated clusters.

**DBSCAN Clustering**

* **Performance:**
  + The silhouette score for **non-scaled data (0.6867)** is higher than that for **scaled data (0.58175)**.
  + This suggests that DBSCAN, like the other two, performs better without scaling.
* **Justification:**
  + DBSCAN identifies clusters based on density, and scaling often improves its robustness.
  + The **epsilon (EPS = 1.4) and min\_samples (5)** show that DBSCAN is identifying dense regions, but the silhouette score drop after scaling suggests that scaling disrupts density-based separations.

**3. Effect of Scaling on Clustering Performance**

From the results, scaling **decreased silhouette scores for all clustering methods**, suggesting that the original feature distribution of the Iris dataset was more effective for clustering. Typically:

* **K-Means and Hierarchical Clustering** are highly sensitive to distance calculations, so scaling is usually expected to improve performance when features have significantly different scales. However, in this dataset, scaling negatively impacted clustering quality.
* **DBSCAN**, which groups data points based on density, usually benefits from scaling, but in this case, the performance dropped slightly.

The reason why scaling negatively affects performance in this scenario might be:

* The **Iris dataset already has a well-defined feature distribution** that naturally separates clusters.
* Scaling **alters the relative distances between points**, which may distort natural clusters.

**Final Conclusion:**

* **Best approach:** **Hierarchical clustering (Non-Scaled)**
* **Reason:** It achieved the highest silhouette score (**0.6867**) and preserved natural clusters.
* **Scaling is not beneficial** in this case, as it reduced clustering performance for all methods

**Effect of Scaling on Clustering**

1. **Scaling Changes Distance Relationships**
   * Many clustering techniques (K-Means, Hierarchical) rely on **distance-based calculations** (e.g., Euclidean distance).
   * Scaling transforms feature values (e.g., by standardization or normalization), altering the **original relative distances** between points.
   * If one feature had a naturally larger range before scaling, it would dominate the clustering. After scaling, this effect is removed, but it may disrupt the **natural structure** in the data.
2. **Loss of Natural Cluster Structure**
   * In datasets like **Iris**, features (sepal/petal length and width) have meaningful scales.
   * Scaling **removes** the original scale relationships, which can cause **overlap between clusters**.
   * If clusters were naturally well-separated before scaling, forcing all features to the same range can make them **artificially closer**, reducing the silhouette score.

Notes:

* The **Iris dataset has three natural clusters** (Setosa, Versicolor, Virginica), but clustering methods found **only two clusters**.
* This may indicate that two species (e.g., Versicolor and Virginica) are **overlapping**, while Setosa remains distinct.
* **Scaling disrupts natural separation**, as petal width and length originally differentiate clusters well.

**Customer Dataset:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Kmeans** | | **Hierarchical** | | **DBScan** | |
| Scaled | Non-Scaled | Scaled | Non-Scaled | Scaled | Non-Scaled |
| **Number of Clusters** | 8 | 2 | 6 | 2 | 4 | 2 |
| **Silhouette Score** | 0.4392 | 0.5834 | 0.4240 | 0.6858 | 0. 3540 | -0.516 |
| **Distance Threshold** | - | - | 3.800007 | 54000 | - | - |
| **EPS** | - | - | - | - | 0.7 | 96.1 |
| **Min Samples** | - | - | - | - | 20 | 5 |

**Conclusion:**

🡪 **Scaling negatively affects K-Means and Hierarchical Clustering but significantly improves DBScan.**

🡪**Hierarchical Clustering is the best choice for non-scaled data due to its highest silhouette score (0.6858).**

🡪**DBScan performs well only when scaling is applied, making it ideal for datasets with arbitrary cluster shapes.**

🡪**K-Means is inconsistent, producing too many clusters when scaled and performing better when non-scaled.**

🡪**Distance-based algorithms like Hierarchical Clustering and DBScan are highly sensitive to scaling.**

**General Notes:**

**K-Means Clustering**

* **Working Principle**: K-Means clusters data by **minimizing the sum of squared Euclidean distances** from points to their cluster centers.
* **Distance Metric**: Relies on **Euclidean distance**, which is sensitive to feature magnitudes.
* **Cluster Shape Assumption**: Assumes clusters are **spherical and evenly sized**.
* **Weaknesses**:
  + **Cannot handle non-spherical clusters**.
  + **Bad at detecting noise and outliers**.
  + **Highly affected by feature scaling**.

**Step-by-Step Explanation of K-Means++**

Imagine we have a dataset with **many points** and we want to cluster them into **K=3 groups**.

**Step 1: Pick the First Centroid Randomly**

* We randomly select **one** point from the dataset as the **first centroid**.

**Step 2: Compute Distance for Each Point**

* For each remaining point **(x)** in the dataset, compute the **distance (D)** to the closest selected centroid.
* If a point is **far from the chosen centroid**, it has a **higher probability of being chosen** as the next centroid.

**Step 3: Pick the Next Centroid Using Probability**

* Instead of choosing randomly, we pick a new centroid **with a higher probability for distant points**.
* The **farther** a point is from an already chosen centroid, the **more likely** it is to be picked.

**Questions:**

**Which clustering approach have you decided to use on each dataset?**

**MultiBlob Dataset**: K-means++ is the best choice here because the dataset is originally made up of 6 clusters, and the method’s output aligns perfectly with this structure. It also has the highest silhouette score, indicating well-separated and well-defined clusters.

**Iris Dataset: Best** approach: Hierarchical clustering (non-scaled)

**Customer Dataset**: Hierarchical Clustering is the best choice for non-scaled data

**Compare between Kmeans, Hierarchical and DBSCAN**

A screenshot of a computer screen

AI-generated content may be incorrect.

**MultiBlob Dataset:**

Originally composed of 6 clusters and Using k-means++ for centroid initialization helps avoid poor clustering by spreading initial centroids apart, leading to more stable results and faster convergence.

The comparison shows that K-Means++ generally leads to better clustering and a higher silhouette score.

**Effect of Different Distance Functions in K-Means, DBSCAN, and Hierarchical Clustering on the Calculated Clusters and Silhouette Score**

Distance functions play a crucial role in clustering because they determine how similarity between data points is measured. Different distance metrics can lead to significant variations in cluster formation, affecting cluster compactness, separation, and overall clustering quality. Below is a detailed breakdown of the impact of different distance functions on **K-Means**, **DBSCAN**, and **Hierarchical Clustering**, along with their effect on the **silhouette score**.

**1. Effect in K-Means Clustering**

K-Means relies on distance measures to assign points to the nearest centroid. The most commonly used metric is **Euclidean distance**, but other metrics can be used.

**Common Distance Metrics and Their Effects:**

1. **Euclidean Distance (L2 Norm)**
   * Measures the straight-line distance between two points.
   * Forms spherical clusters.
   * Works best when clusters are convex and isotropic.
   * **Effect on silhouette score**: Works well if the dataset has clear circular clusters, but performs poorly for elongated or irregularly shaped clusters.

**Effect in Hierarchical Clustering (Agglomerative & Divisive)**

Hierarchical clustering uses distance metrics in combination with **linkage criteria** to determine which clusters to merge.

**Common Distance Metrics and Their Effects:**

1. **Euclidean Distance**
   * Forms spherical clusters.
   * Works well with **Ward linkage** (which minimizes variance).
   * **Effect on silhouette score**: High when data naturally forms spherical clusters.
2. **Manhattan Distance**
   * Forms grid-aligned clusters.
   * Can be useful in structured data.
   * **Effect on silhouette score**: May lower score if clusters are not grid-aligned.
3. **Cosine Similarity**
   * Groups points with similar directional vectors.
   * Works well with text or high-dimensional data.
   * **Effect on silhouette score**: Can be high for text-based clustering.

**Effect of Linkage Criteria in Hierarchical Clustering**

Linkage methods affect how distances between clusters are computed:

1. **Single Linkage**
   * Uses minimum pairwise distance.
   * Produces elongated, chain-like clusters.
   * **Effect on silhouette score**: Can be lower due to long chains.
2. **Complete Linkage**
   * Uses maximum pairwise distance.
   * Produces compact, well-separated clusters.
   * **Effect on silhouette score**: Usually improves score if clusters are well-separated.
3. **Average Linkage**
   * Uses mean pairwise distance.
   * Balances between single and complete linkage.
   * **Effect on silhouette score**: Generally moderate performance.

**2. Effect in DBSCAN (Density-Based Clustering)**

DBSCAN clusters based on **density** rather than distance from a centroid, so the choice of distance metric directly affects how neighborhoods are defined.

**Common Distance Metrics and Their Effects:**

1. **Euclidean Distance**
   * Finds compact, dense clusters.
   * Sensitive to varying densities.
   * **Effect on silhouette score**: High if clusters are naturally dense; low if clusters have varying densities.

**NOTE:**

🡪 The K-distance graph gives a **starting point** for eps.

🡪 The **silhouette score** finds the best combination of eps and MinPts for **well-separated** clusters.

🡪The values may differ because **density estimation and optimal clustering shape are different goals**.

**Why Does the MinPts from the K-Distance Graph Not Match the Optimal Eps Later?**

1. **K-Distance Graph Focuses on Global Density Estimation**
   * The K-distance graph helps find a global density threshold (eps), but this might not be the best for actual clustering performance.
   * The silhouette score, on the other hand, measures how well-separated clusters are, which could favor a different combination of eps and MinPts.
2. **Eps is Sensitive to Data Distribution**
   * The "elbow" in the K-distance graph is just an estimate of the **natural density break** in your data.
   * However, in actual clustering, slightly different eps might improve separability.