NAME: Sahil Motiramani DIV:D15C ROLL NO:35

**AIDS Lab Exp 07**

**Aim**: To implement different clustering algorithms.
**Theory**: Clustering is an unsupervised machine learning technique used to group similar data points together. The objective is to discover natural groupings within a dataset without prior knowledge of class labels. It is widely applied in fields such as:

- Customer segmentation
- Anomaly detection
- Image segmentation
- Bioinformatics

## Types of Clustering:

1. **Partition-based Clustering (e.g., K-Means)**
    - Divides data into a predefined number of clusters.
    - Each data point belongs to exactly one cluster.
    - Example Algorithm: K-Means
2. **Density-based Clustering (e.g., DBSCAN)**
    - Forms clusters based on dense regions in the data.
    - Can identify clusters of arbitrary shape and detect noise (outliers).
    - Example Algorithm: DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
3. **Hierarchical Clustering**
    - Builds a tree of clusters using:
        - Bottom-up approach (Agglomerative) – Start with individual points and merge clusters.
        - Top-down approach (Divisive) – Start with a single cluster and split it progressively.
    - Example Algorithm: Agglomerative Clustering
4. **Model-based Clustering**
    - Assumes data is generated by a mixture of underlying probability distributions.
    - Fits a probabilistic model to the data to identify clusters.
    - Example Algorithm: Gaussian Mixture Models (GMM)

## K-Means Clustering

K-Means is a partition-based clustering algorithm that divides data into K clusters. It aims to minimize the intra-cluster variance by assigning each data point to the nearest centroid.

---

## Algorithm Steps:

1. Choose the number of clusters (K).
2. Randomly initialize K centroids (initial cluster centers).
3. Assign each data point to the nearest centroid (based on Euclidean distance).
4. Update centroids by computing the mean of all points in each cluster.
5. Repeat steps 3 & 4 until convergence (when centroids stop changing significantly or a maximum number of iterations is reached).

**Key Considerations:**

- Use the Elbow Method to find the point where adding more clusters yields diminishing returns.
- Use the Silhouette Score to measure how well-separated and cohesive the clusters are.

## Agglomerative Clustering

Agglomerative Clustering is a hierarchical clustering algorithm that uses a bottom-up approach. It starts by treating each data point as its own cluster and iteratively merges the closest clusters until a single cluster remains or a predefined number of clusters is achieved.

---

## Algorithm Steps:

1. Start with each data point as an individual cluster.
2. Compute the distance (similarity) between all clusters.
3. Merge the two closest clusters.
4. Update the distance matrix to reflect the new cluster.
5. Repeat steps 2–4 until:
    - A single cluster remains (full dendrogram), OR
    - A predefined number of clusters is reached.

**Step 1. Load and preprocess the dataset.**

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender   Age  \
0          1    15634602  Hargrave          619    France  Female  42.0
1          2    15647311      Hill          608     Spain  Female  41.0
2          3    15619304      Onio          502    France  Female  42.0
3          4    15701354      Boni          699    France  Female  39.0
4          5    15737888  Mitchell          850     Spain  Female  43.0

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2   74274.87              1   1.000000             1.0
1       1   83807.86              1   0.000000             1.0
2       8  159660.80              3   1.000000             0.0
3       1  117561.49              2   0.000000             0.0
4       2  125510.82              1   0.705529             1.0

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

Useful Features for Clustering:

- **CreditScore**: Reflects financial health.
- **Age**: Important for segmenting by age groups.
- **Balance**: Indicates customer wealth.
- **NumOfProducts**: Measures engagement level.
- **EstimatedSalary**: Income distribution is relevant.
- **Geography and Gender**: Can be encoded for segmentation.

Irrelevant Features for Clustering:

- **RowNumber**, CustomerId, Surname: Unique identifiers, not useful.
- **Exited**: A target variable for classification, not used in clustering.

**Step 2. Elbow method for number of clusters**

The Elbow Method plot helps determine the optimal number of clusters (K) in K-Means by plotting inertia (sum of squared distances to cluster centers) against different K values. The "elbow" point, where inertia reduction slows significantly, indicates the ideal K.
Formula:

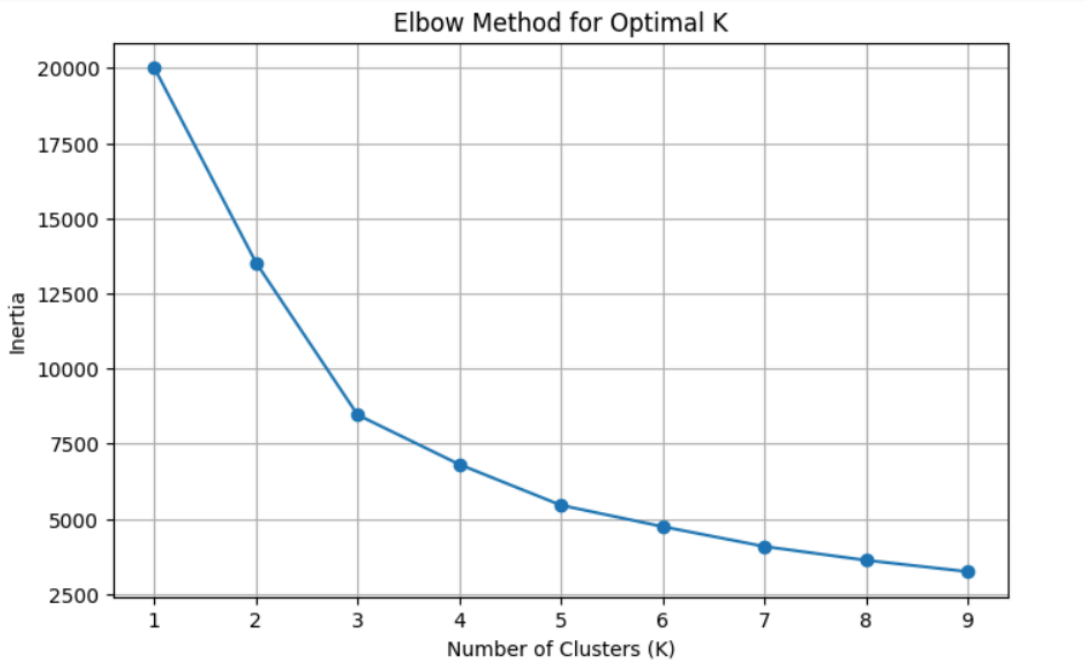
NAME: Sahil Motiramani DIV:D15C ROLL NO:35

$$WCSS = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

Where,

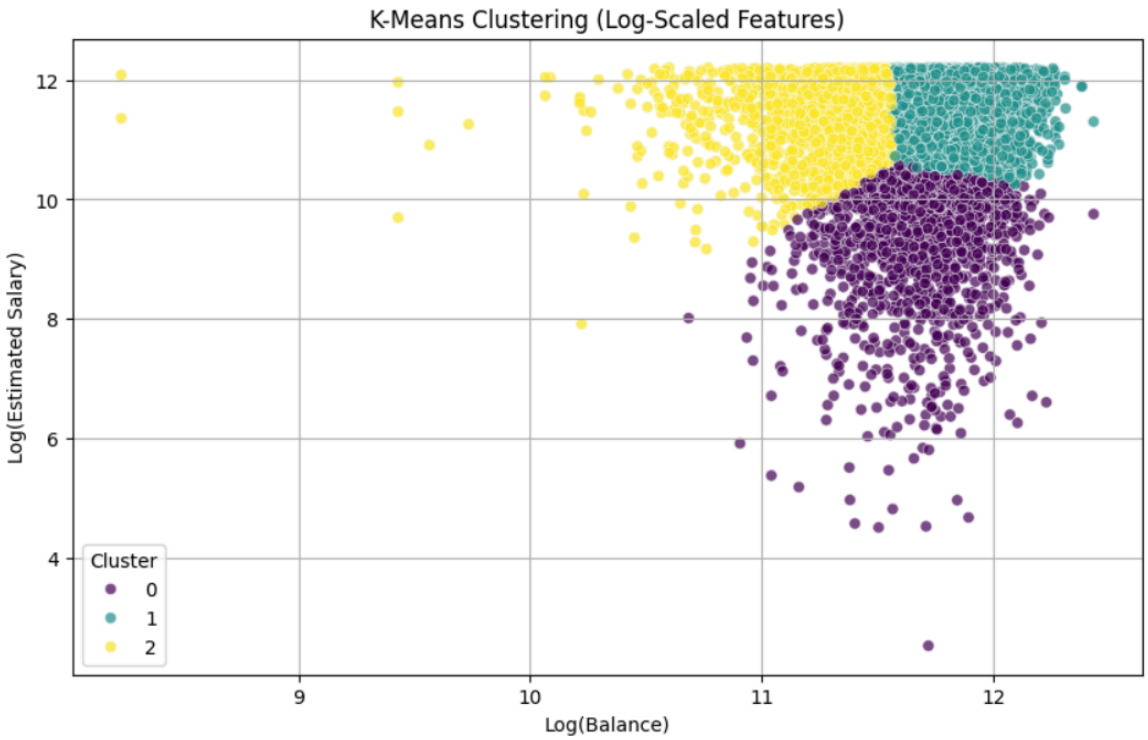
- Ci = Cluster i
- μi = Centroid of cluster Ci
- || xi - μi ||² = Squared Euclidean distance between a point and its cluster centroid

```
# Apply log transformation to skewed features (to reveal better patterns)
df['log_balance'] = np.log1p(df['Balance'])
df['log_salary'] = np.log1p(df['EstimatedSalary'])

# Select improved features for clustering
features = df[['log_balance', 'log_salary']]

# Scale the features for K-Means
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Elbow Method to find optimal K
distortions = []
K_range = range(1, 10)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(features_scaled)
    distortions.append(kmeans.inertia_)

# Plot Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(K_range, distortions, marker='o', linestyle='-')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.grid(True)
plt.show()
```




This plot shows the Elbow Method, which helps determine the optimal number of clusters (K) for K-Means clustering. The "elbow point" represents where the inertia (sum of squared distances) stops decreasing significantly. In this case, the elbow is around **K=3**, indicating three clusters is a suitable choice.

```
# Apply K-Means Clustering
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
df['kmeans_cluster'] = kmeans.fit_predict(features_scaled)

# Visualize K-Means Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=df['log_balance'],
    y=df['log_salary'],
    hue=df['kmeans_cluster'],
    palette='viridis',
    alpha=0.7
)
plt.xlabel('Log(Balance)')
plt.ylabel('Log(Estimated Salary)')
plt.title('K-Means Clustering (Log-Scaled Features)')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```




This plot visualizes the K-Means clustering results on log-scaled features (Balance and Estimated Salary). Three distinct clusters are identified, each representing groups with similar financial profiles. Different colors represent different clusters, highlighting how K-Means groups customers based on these two variables.

## DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

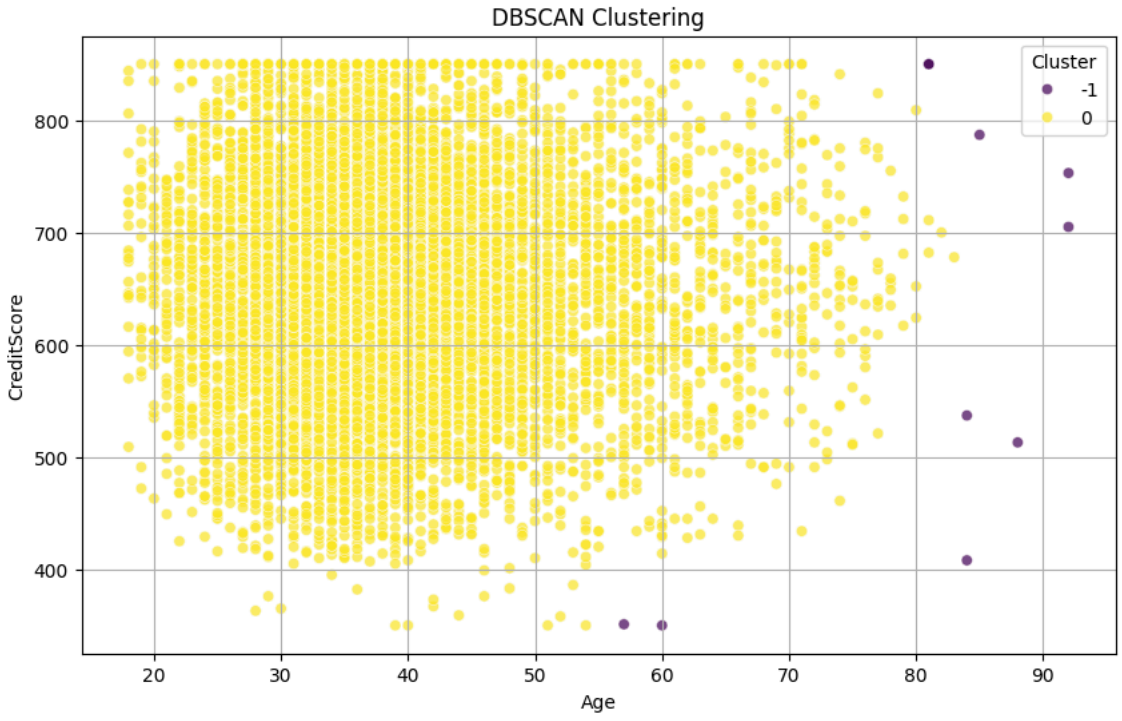
DBSCAN is a powerful clustering algorithm that groups together points that are densely packed and identifies outliers as noise. Unlike K-Means, it does not require specifying the number of clusters in advance and can detect clusters of arbitrary shapes.

DBSCAN relies on two key parameters:

1. **Epsilon(eps)** – The radius within which points are considered neighbors.
2. **MinPts** – The minimum number of points required to form a dense region (cluster).

NAME: Sahil Motiramani DIV:D15C ROLL NO:35

```
# Select features (without log transformation)
features = df[['Age', 'CreditScore']]

# Scale features (Standardizing helps DBSCAN performance)
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=10)
df['dbscan_cluster'] = dbscan.fit_predict(features_scaled)

# Inspect clusters
print(df['dbscan_cluster'].value_counts())

# Visualize DBSCAN Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=df['Age'],
    y=df['CreditScore'],
    hue=df['dbscan_cluster'],
    palette='viridis',
    alpha=0.7
)
plt.xlabel('Age')
plt.ylabel('CreditScore')
plt.title('DBSCAN Clustering')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



## Agglomerative Hierarchical Clustering:

Agglomerative Hierarchical Clustering is a bottom-up approach that starts by treating each data point as its own cluster and progressively merges the closest clusters until a desired number of clusters is reached. For your dataset, which involves customer information, this method helps to group customers based on attributes like **credit score** and **balance**. It is useful for customer segmentation, identifying patterns, and understanding customer behavior.

**Key Steps:**

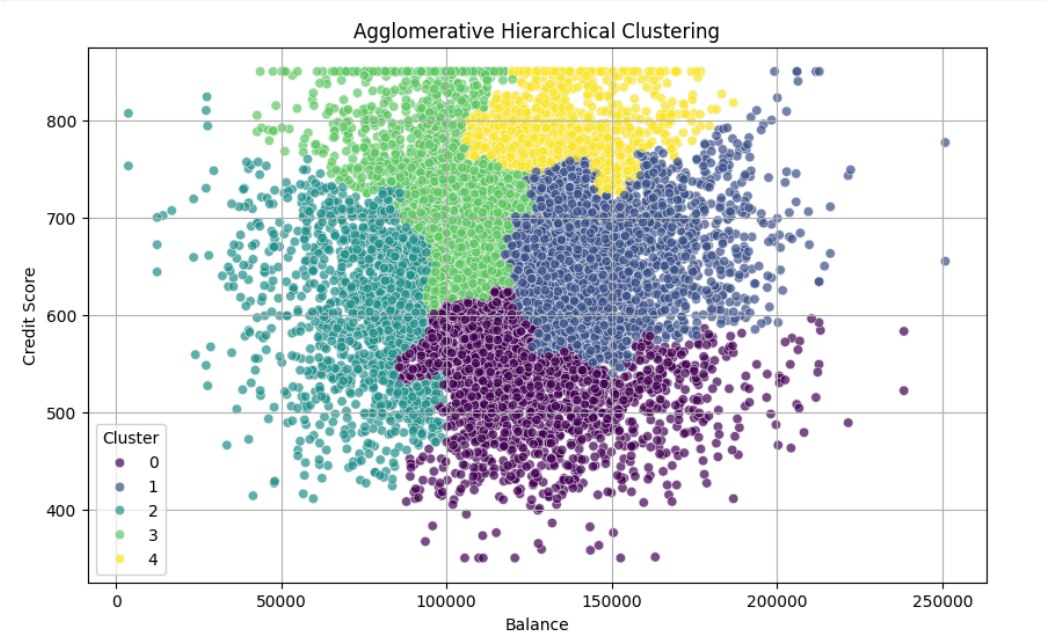
1. Each data point starts as a separate cluster.
2. Iteratively merge the two closest clusters.
3. Continue merging until the desired number of clusters is achieved

**Advantages:**

- No need to specify the number of clusters in advance.
- Suitable for capturing hierarchical relationships in data.

```
from sklearn.cluster import AgglomerativeClustering
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler

# Scale features (for better clustering performance)
scaler = StandardScaler()
features_scaled = scaler.fit_transform(df[['Balance', 'CreditScore']])

# Create Agglomerative Clustering model
agg_cluster = AgglomerativeClustering(n_clusters=5, linkage='ward')
df['agg_cluster'] = agg_cluster.fit_predict(features_scaled)

# Visualize Agglomerative Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=df['Balance'],
    y=df['CreditScore'],
    hue=df['agg_cluster'],
    palette='viridis',
    alpha=0.7
)
plt.xlabel('Balance')
plt.ylabel('Credit Score')
plt.title('Agglomerative Hierarchical Clustering')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```

NAME: Sahil Motiramani DIV:D15C ROLL NO:35




## Graph Interpretation (Agglomerative Clustering Visualization)

1. **Axes**: The x-axis represents the Balance (bank account balance), and the y-axis represents the Credit Score of customers.
2. **Clusters**: The plot shows five distinct clusters (0, 1, 2, 3, 4) based on these two features.
3. **Cluster Distribution**:
    - Cluster 0 (purple) represents customers with low credit scores and mid-to-high balances.
    - Cluster 1 (blue) contains moderate credit scores and higher balances.
    - Cluster 2 (cyan) consists of low-to-moderate credit scores and low balances.
    - Cluster 3 (green) includes higher credit scores and mid-level balances.
    - Cluster 4 (yellow) represents customers with the highest credit scores and highest balances.
4. **Insight**: This clustering helps identify customer segments, such as high-value customers, low-credit-risk customers, or those at potential risk of churn.
5. **Application**: Useful for targeted marketing strategies and risk assessment by identifying which groups need specific attention.
6. **Diversity of Groups**: Each cluster reflects unique customer behaviors, assisting in better decision-making for customer retention or personalized offerings.

## Silhouette Score in Clustering

NAME: Sahil Motiramani DIV:D15C ROLL NO:35

The Silhouette Score is a metric used to evaluate the quality of clustering in an unsupervised learning context. It measures how well each data point is clustered by comparing its cohesion (how close it is to other points in the same cluster) and its separation (how far it is from points in other clusters).

Formula:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- a(i)a(i)a(i) = The average intra-cluster distance (cohesion)
  - This is the average distance between iii and all other points in the same cluster.
- b(i)b(i)b(i) = The average inter-cluster distance (separation)
  - This is the average distance between iii and the nearest cluster to which it does not belong.

```python
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Scale features for better performance
scaler = StandardScaler()
features_scaled = scaler.fit_transform(df[['Balance', 'CreditScore']])

# Apply K-Means
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['kmeans_cluster'] = kmeans.fit_predict(features_scaled)

# Calculate Silhouette Score
kmeans_silhouette = silhouette_score(features_scaled, df['kmeans_cluster'])
print(f"K-Means Silhouette Score: {kmeans_silhouette:.3f}")
```

```
K-Means Silhouette Score: 0.316
```

```python
from sklearn.cluster import AgglomerativeClustering

# Apply Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=4, linkage='ward')
df['agglo_cluster'] = agglo.fit_predict(features_scaled)

# Calculate Silhouette Score
agglo_silhouette = silhouette_score(features_scaled, df['agglo_cluster'])
print(f"Agglomerative Clustering Silhouette Score: {agglo_silhouette:.3f}")
```

```
Agglomerative Clustering Silhouette Score: 0.260
```

The **Silhouette Score** ranges from **-1 to 1**, where:

- **+1** → Well-clustered data (points are close to their own cluster and far from others).
- **0** → Overlapping clusters (data points are on the boundary between clusters).
- **-1** → Misclassified points (closer to a different cluster than their own).

**K-Means performed better than Agglomerative Clustering, based on the Silhouette Score (0.316 vs. 0.260).**

If cluster shapes are **non-spherical**, other methods like **DBSCAN** or different linkages (e.g., complete, average) in Agglomerative Clustering might improve results.

**Conclusion**

This experiment applies clustering techniques to a bank churn model, segmenting customers based on financial behavior such as balance and credit score. K-Means Clustering effectively groups customers into predefined clusters, helping banks identify those at risk of churning, while DBSCAN detects dense regions of similar customers and outliers without requiring a predefined number of clusters. Additionally, Agglomerative Clustering provides a hierarchical approach, capturing different levels of customer similarity to enhance retention strategies. The Silhouette Score evaluates clustering performance, ensuring well-separated and meaningful groups. By leveraging these techniques, banks can improve customer retention, offer personalized financial products, and optimize marketing efforts.