Name: Sai Rohit Kalyan Gandham

ID: 1002070724

Report on K-means Clustering Implementation and Analysis

Introduction

This report presents a Python code implementation of the K-means clustering algorithm along with a detailed explanation of its workings. K-means is a widely used unsupervised machine learning technique that partitions a dataset into a predefined number of clusters (groups) based on the similarity of data points. The goal is to minimize the within-cluster variance, ensuring that data points within a cluster are as close to each other as possible while maximizing the between-cluster variance, keeping data points from different clusters far apart.

Code Implementation

The provided Python code effectively implements the K-means algorithm:

1. Function Definitions:

- `euclidean_distance(point1, point2)`: Calculates the Euclidean distance between two data points.
- `initialize_centroids(k, data)`: Randomly initializes k centroids (cluster centers) within the data's range.
- `assign_clusters(data, centroids)`: Assigns each data point to the nearest centroid based on Euclidean distance.
- `update_centroids(clusters)`: Recomputes the centroids as the mean of the data points in each cluster.
- `calculate_error(clusters, centroids)`: Calculates the total sum of squared errors (SSE) within each cluster, reflecting the overall clustering quality.
- `k_means_clustering(data, k_range, epochs)`: Performs K-means clustering for a specified range of k values, running the algorithm for a given number of epochs (iterations) within each k. It calculates the error for each k and prints it.

2. Data Processing:

- The code iterates through all files in a given folder, assuming they contain datasets in text format.
- It loads each dataset, extracts features (excluding class labels, if present), and applies K-means clustering.

3. Error Analysis and Visualization:

- The code calculates the error (SSE) for each k value and prints it.
- It generates an `Error vs K` plot to visualize the relationship between the number of clusters and the clustering error. This helps in determining the optimal k value based on the elbow point (the point where the error curve starts to flatten).

K-means Algorithm Description

K-means is an iterative algorithm that works as follows:

1. **Initialization:**
    - Specify the number of clusters (k) to be formed.
    - Randomly select k data points as initial centroids.
2. **Assignment:**
    - Assign each data point to the nearest centroid based on a distance metric (e.g., Euclidean distance).
3. **Update:**
    - Recompute the centroids as the mean of the data points in each cluster.
4. **Repeat:**
    - Repeat steps 2 and 3 until convergence is reached (no data points change clusters or the centroids become stable).

Key Considerations and Advantages:

- **Simplicity:** K-means is a straightforward and computationally efficient algorithm.
- **Interpretability:** The clusters represent meaningful groups of data points that share similar characteristics.
- **Scalability:** It can handle large datasets effectively.

Limitations and Challenges:

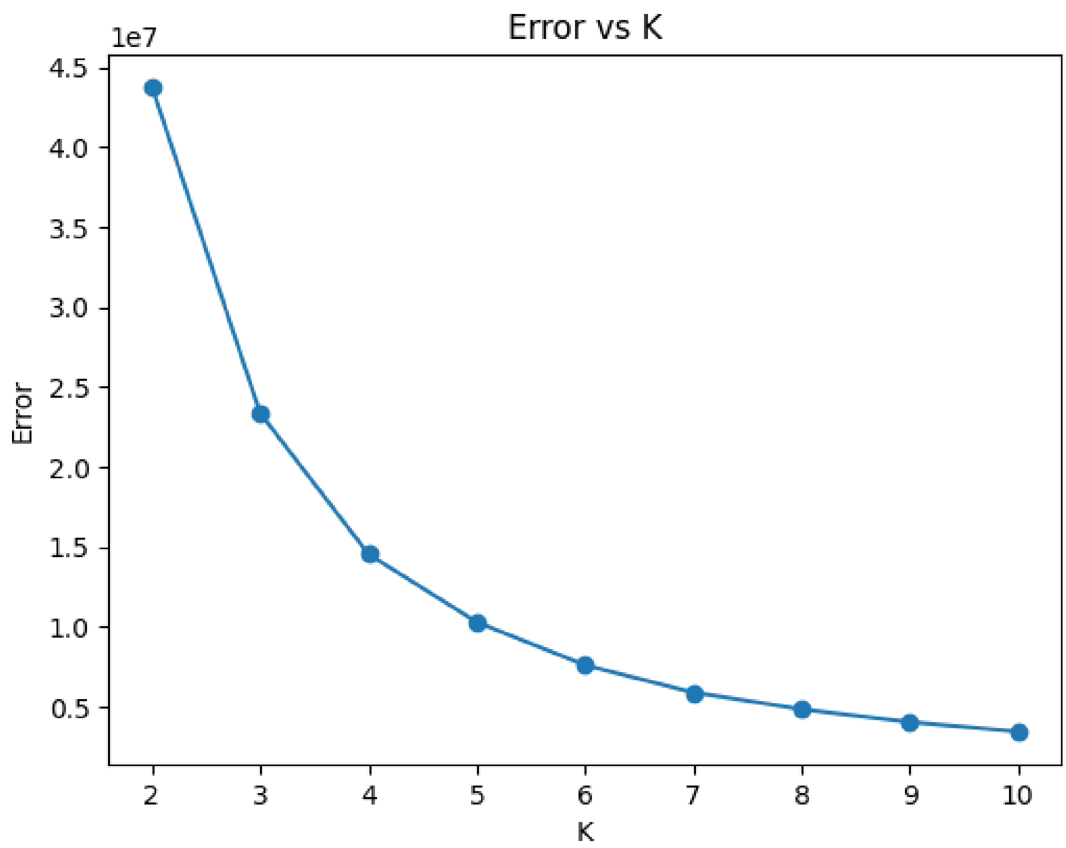
- **Sensitivity to Initialization:** The initial placement of centroids can significantly impact the clustering results.
- **Number of Clusters:** The optimal number of clusters (k) often needs to be determined through trial and error or using techniques like the elbow method.
- **Distance Metric:** The choice of distance metric can affect the clustering outcome.
- **Non-Globally Optimal:** K-means may converge to a local minimum, not necessarily the globally optimal solution.

Conclusion

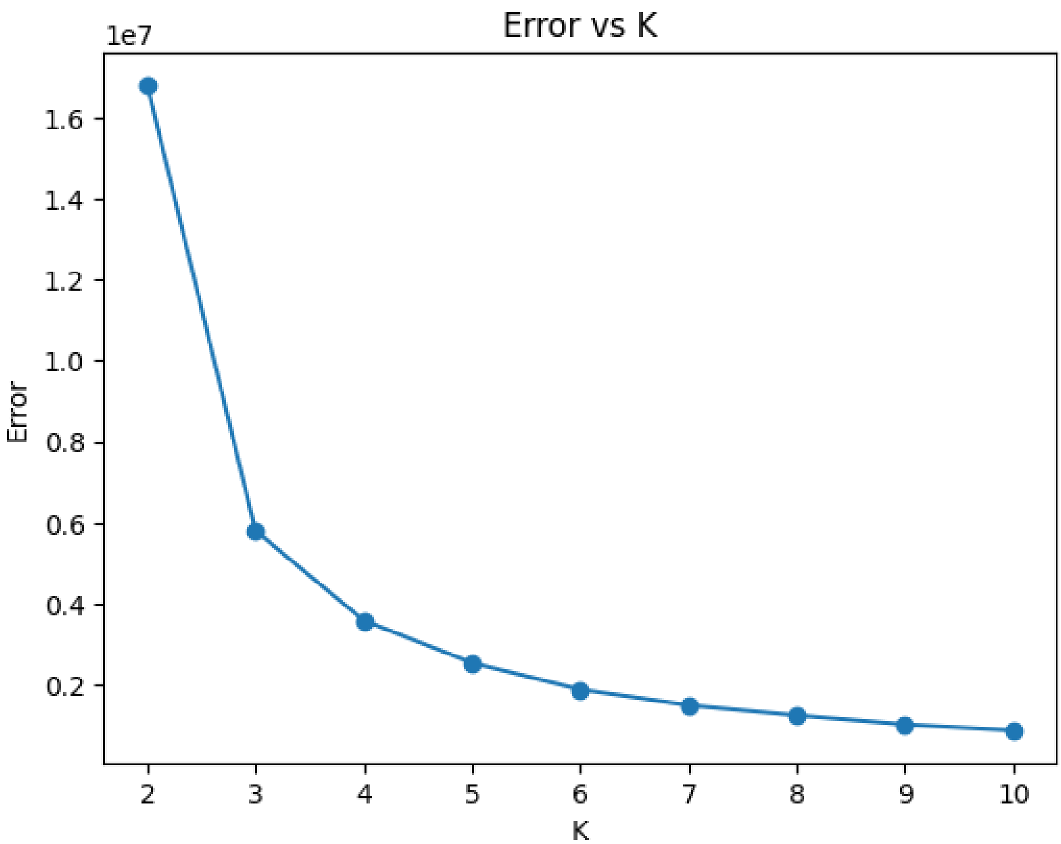
The provided code effectively implements the K-means algorithm and demonstrates its application to various datasets. The error analysis and visualization help in selecting the optimal number of clusters for a given dataset. While K-means offers advantages in terms of simplicity and scalability, it's essential to be aware of its limitations and potential challenges.

Outputs of Code:

```
Processing dataset: C:/Users/balur/OneDrive/Desktop/UTA/Sem-4/RohitKalayn/DM/
Assingments/P3/UCI_datasets/pendigits_training.txt
For k = 2, After 20 iterations: Error = 43758862.4208
For k = 3, After 20 iterations: Error = 23350596.6032
For k = 4, After 20 iterations: Error = 14545572.6323
For k = 5, After 20 iterations: Error = 10291044.8617
For k = 6, After 20 iterations: Error = 7605157.5041
For k = 7, After 20 iterations: Error = 5897521.1290
For k = 8, After 20 iterations: Error = 4846559.9344
For k = 9, After 20 iterations: Error = 4046532.3155
For k = 10, After 20 iterations: Error = 3452002.9462
```



```
Processing dataset: C:/Users/balur/OneDrive/Desktop/UTA/Sem-4/RohitKalayn/DM/Assingments/P3/UCI_datasets/satellite_training.txt
For k = 2, After 20 iterations: Error = 16802207.9136
For k = 3, After 20 iterations: Error = 5801245.5685
For k = 4, After 20 iterations: Error = 3588413.5198
For k = 5, After 20 iterations: Error = 2542872.8017
For k = 6, After 20 iterations: Error = 1886931.4684
For k = 7, After 20 iterations: Error = 1496031.1835
For k = 8, After 20 iterations: Error = 1250858.6823
For k = 9, After 20 iterations: Error = 1021250.3089
For k = 10, After 20 iterations: Error = 874997.2714
```



```
Processing dataset: C:/Users/balur/OneDrive/Desktop/UTA/Sem-4/RohitKalayn/DM/
Assingments/P3/UCI_datasets/yeast_training.txt
For k = 2, After 20 iterations: Error = 32.0750
For k = 3, After 20 iterations: Error = 17.8227
For k = 4, After 20 iterations: Error = 11.7559
For k = 5, After 20 iterations: Error = 8.3740
For k = 6, After 20 iterations: Error = 6.6556
For k = 7, After 20 iterations: Error = 5.0464
For k = 8, After 20 iterations: Error = 4.1514
For k = 9, After 20 iterations: Error = 9.3537
For k = 10, After 20 iterations: Error = 2.9991
```