International Journal of Computer Sciences and Engineering
Vol.**13**, Issue.**4**, pp.**68-77** April **2025**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Article

Enhanced K-Means Clustering through Density-Based Inter-Centroid Distance Optimization

Snehal K Joshi[1*]

[1]Dept. of Computer, Dolat-Usha Institute of Applied Sciences, Affiliated to Veer Narmad South Gujrat University, Valsad, India

Corresponding Author:

Received: 20/Feb/2025; **Accepted**: 21/Mar/2025; **Published**: 30/Apr/2025. **DOI**: https://doi.org/10.26438/ijcse/v13i4.6877

Abstract: K-Means is a popular algorithm used in unsupervised machine learning for clustering tasks, especially when working with data that lacks predefined labels. It is widely employed to divide such data into meaningful groups. This research presents an improved version of the traditional K-Means algorithm, adapting it for use on labeled datasets to evaluate its effectiveness in segmentation. The study compares this modified version with the standard K-Means method, focusing on aspects like accuracy, efficiency, and computational demand. A dataset containing more than 3,000 records is used for experimentation. The standard approach starts with K=2, randomly selecting initial centroids and refining them through iterations until results stabilize. This is repeated up to K=9. In the revised method, however, a top-down approach is implemented. Instead of selecting centroids randomly, the algorithm uses a density-based technique to place initial centroids in densely populated data regions. Clusters are formed based on these regions and refined iteratively. After each convergence, the process continues by further dividing the clusters, up to K=9. Results from the study reveal that the new approach improves performance by speeding up convergence—reducing iterations by over 20%—and lowering computational costs, while also boosting overall clustering accuracy and efficiency.

Keywords: Clustering, K-Means Algorithm, Density-Based Centroid Selection, Top-Down Approach, Segmentation Efficiency

1. Introduction

Clustering is a technique used to organize data points into groups based on similarities in their features. Generally, data analysis involves two primary approaches depending on whether the dataset is labeled or unlabeled. For labeled datasets, data is grouped according to specific features, and these groupings are used to train a model, which is then validated against a test set containing known outcomes, or ground truth. This process falls under classification, a supervised learning method where models such as Polynomial Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) are employed. These models operate on the assumption that the number of output classes is predetermined.

On the other hand, when dealing with unlabeled datasets, where the model cannot be trained using predefined outcomes, clustering becomes the preferred method. It identifies natural groupings in the data by measuring similarity or proximity among data points, without relying on any prior labeling. Since the exact number of groups is not initially defined, clustering is classified as an unsupervised learning technique.

Among clustering algorithms, K-Means is widely regarded for its simplicity and effectiveness. It functions by forming K groups based on feature similarity, although the optimal value of K is often not known in advance. The structure of the resulting clusters is influenced by the underlying characteristics of the dataset.

1.1 Functionality of K-means Algorithm

The K-Means clustering algorithm follows a structured process that can be outlined in five main steps. Clusters are formed based on the inherent features of the dataset. To initiate clustering, one of two methods is typically used for selecting the initial centroids. In the first method, k data points are randomly chosen from the dataset to serve as centroids. Alternatively, the second method involves generating k new points that act as initial centroids.

In the first step, the algorithm begins by assigning these k centroids. The second step involves computing the distance

between each data point and every centroid. This distance can be measured using various dissimilarity functions such as Euclidean distance, Minkowski distance, or cosine similarity. A distance matrix is then created representing the closeness of all data points to each centroid.

The third step clusters each data point with the nearest centroid based on the calculated distances. These associations form preliminary clusters. In the fourth step, the centroid of each cluster is updated by calculating the average position of all the points in the cluster, often using Mean Squared Error (MSE) to reposition centroids more accurately.

The fifth step involves repeating steps 2 through 4 until the centroids stabilize, meaning they no longer change significantly between iterations — a state referred to as convergence. Once convergence is reached for a specific k, the value of k is increased, and the entire five-step process is repeated. This iterative process continues until convergence is achieved for each incremented value of k, refining the clustering outcome progressively.

1.2 Selection of k- value

Determining the optimal number of clusters (k) is a crucial aspect of implementing the K-Means algorithm, as it directly influences the quality and accuracy of the clustering results. For each value of *k*, there are *k* centroids assigned, and evaluating their suitability requires analyzing two primary distance metrics:
(i) Intra-cluster distance, which measures how close individual data points are to the centroid of the cluster they belong to, and
(ii) Inter-cluster distance, which calculates the distance between different cluster centroids.

Given a dataset of *n* data points, when segmented into *k* clusters, it becomes necessary to compute the distance d between each data point and all *k* centroids. This means for every data point dp_1 *to* dp_n, the distance is calculated relative to centroids C_1 *to* C_k as shown in (1). These values are used to create a two-dimensional distance matrix, where each row corresponds to a data point and each column corresponds to a centroid.

This matrix can be denoted as $d_{1..n} \times c_{1..k}$, where the entry at each cell represents the distance from a specific data point to a particular centroid. The matrix provides a comprehensive view of the proximity of data points to all cluster centers and is foundational for both assigning data points to clusters and evaluating clustering performance through intra- and inter-cluster analysis.

$$\begin{bmatrix} dp1(c=1) & \cdots & dp1(c=k) \\ \vdots & \ddots & \vdots \\ dpn(c=1) & \cdots & dpn(c=k) \end{bmatrix} \quad (1)$$

1.3 Performance measurement of Algorithm

When evaluating the performance of the K-Means clustering algorithm, two critical computational factors must be considered: time complexity and space complexity. These metrics help in understanding the algorithm's efficiency, especially when dealing with large datasets.

For a dataset containing n data points with m features or dimensions, the complexity of K-Means depends on the number of clusters (k) and iterations (i) required for convergence. The time complexity is primarily influenced by the number of operations needed during each iteration. Specifically, in each iteration, the algorithm must calculate the distance between each data point and all *k* centroids.

Let's assume that the algorithm runs for *i* iterations. During every iteration, for each of the *n* data points, the distance to all *k* centroids is computed using a selected proximity measure. If Euclidean distance is used, then according to its formula, six basic operations are required per distance calculation: two subtractions, two multiplications, one addition, and one square root operation—performed for each feature of a data point in relation to a centroid.

As a result, the time complexity of the K-Means algorithm can be expressed as O(n × k × i × m), considering all distance calculations across iterations. Likewise, the space complexity is mainly affected by the need to store distances, centroids, and intermediate assignments, which also scales with the size of the dataset and the number of clusters. We can represent it as shown in (2).

$$d(x_i, y_i) = \sqrt{(xi - xc)^2 + (yi - yc)^2} \quad (2)$$

For purpose of measuring time complexity, we consider three computations. For i iterations, the time complexity can be measured based on (i) distance calculation using proximity function (Euclidean function in this case), (ii) comparing distances by obtaining k centroids for each data points to identify the nearest centroid and (iii) finally using mean of distance for k clusters in process to obtain new centroid. Hence, for every iteration, total operations can be sum of these three measurements which are
6*[total clusters * total data points * total features],
[(total clusters-1)*total data points * total features] and [total clusters * ((total data points-1) + 1)* total features].

If we consider that the convergence is obtained with i iterations then total operations will be 6*[total iterations *total clusters * total data points * total features], [total iterations *(total clusters -1)*total data points * total features] and [total clusters * ((total data points-1) + 1)* total features] operations. Hence, the time complexity can be measured as O(total iterations*total clusters*total data points*total features). We can represent it as shown in (3).

$$O(i * k * n * m) \quad (3)$$

When working with datasets containing a large number of data points, but relatively fewer clusters and limited features per data point, the computational dynamics shift. In such cases, the general relationship among the variables can be expressed as k << m << n, meaning that the number of clusters (*k*) is significantly smaller than the number of

features (*m*), which is itself much smaller than the number of data points (*n*).
Given this relationship, the time complexity of the K-Means algorithm, previously estimated as:
O (i×k×n×m)\mathcal (where *i* is the number of iterations), can be approximated as:
O (n).
This simplification holds under the assumption that *k*, *m*, and *i* are relatively small constants compared to *n*.
Regarding space complexity, the algorithm needs to store all *n* data points, each with *m* features, as well as *k* cluster centroids, also with *m* features each. Thus, the space required is proportional to:
O ((n+k)×m)
This reflects the combined memory required for storing both the dataset and the centroids during the clustering process.

2. Literature Reviews and Objective

Traditional k-means clustering algorithm is unsupervised algorithm and there are two major short comings with this algorithm. First short coming is selecting k clusters initially. As there is no thumb rule that how many clusters we should go for in progressive way when the algorithm is implemented. Normally, we start with k=2 and select two centroids either we randomly select them or generate two random centroids. Then, we compute the distance matrix and form clusters for given centroids. Again, we compute mean and obtain new centroid for the clusters. This process we continue until convergence is achieved. Once convergence is achieved, we increase the clusters hence the value of k and again repeat the process. In many cases the local minima is achieved instead of having global minima and hence, we do not achieve optimum convergence. As such we cannot predict or determine number of iterations until the convergence is achieved. Similarly, up to which value the k is to be increased is also not fix. Various approaches are used for this purpose. When data set size is very large, measuring intra cluster data point and inter cluster distance is as well as obtaining mean value for each cluster and reshapes the centroid makes the performance of algorithm slow.

To address these short comings, many methods are proposed in process to improve performance of k-means algorithm. As per a study, in process to improve the performance of the algorithm, researchers proposed k numbers of distance tree data structure for each data points [1]. The study proposed to introduce k distance tree data structure to determine the centroids. The approach is easy to implement and reduce the problem of local minima at good extent. Another study which used six data sets and worked on road lanes detection automatically by generating clusters using various features. It is pertaining to these datasets and Global positioning System (GPS) [2]. In another study, the k-mean algorithm is extended for the categorical data domains [3]. The study is pertaining to use of pattern mixing algorithm and study the resultant observations which results in enhancement of results. It can be used to solve real world problems having composite data and higher level of noise. One of the studies on interpretation and validation of cluster analysis, suggest that silhouettes coefficient score is suggestive important measure to analyze clusters obtained. Using silhouettes score, clusters significance can be analyzed [4]. K-mean algorithm is sensitive and performance depends on initialization of centroids. Due to this, it is likely to stuck at local minima rather than achieving global minima and hence, the convergence achieved can be a false convergence. Moreover, outliers can influence the mean of intra cluster Mean Square Error (MSE) and hence, dealing with outliers is also a task. To address these problems, Singh and Gill proposed a modified algorithm which perform with Time Complexity in worst case at $O(n^2)$ compared to $O(n^i)$ where 2<=i<=3 in case of traditional k-means algorithm [5]. One proposed algorithm suggested initialization of centroid instead of having random selection of centroid in cluster. This approach is selective centroid approach and gives better performance [6]. In another study, pattern mixing algorithm proposed to categorical domains like reference journals, topics and subtopics. It provides better accuracy in process to form clusters compared to the traditional k-means algorithm [7]. Clustering algorithms, particularly K-means, are widely employed in the field of machine learning for segmenting data into groups. Several studies have explored the optimization and enhancement of the K-means clustering algorithm to improve its performance and efficiency.

One significant study by Zhang et al. (2016) focused on optimizing the K-means algorithm by proposing a hybrid model combining K-means with a density-based approach. Their method aimed to improve cluster quality and minimize the error rate by adjusting the centroid initialization, which is a common issue in traditional K-means algorithms. They found that their hybrid approach yielded more accurate clusters, particularly in the presence of noise in the datasets. In another study, Gupta and Sharma (2017) explored the use of an adaptive K-means algorithm[8]. They developed a framework that adjusts the value of K during the clustering process. The method dynamically selects the optimal number of clusters based on the distribution of data, which eliminates the need for manual K selection. The results indicated that this adaptive model enhanced the clustering process, especially when dealing with large datasets. Further research by Kim et al. (2018) examined the application of K-means clustering in image segmentation tasks[9]. They modified the traditional K-means algorithm to incorporate a spatial distance component that accounts for pixel location. This modification improved the algorithm's ability to identify clusters in images with complex textures and varied pixel distributions. The study demonstrated that incorporating spatial information led to better segmentation accuracy in image processing applications. Moreover, a study by Lee et al. (2019) proposed a hybrid K-means algorithm that integrated a decision tree to automatically determine the number of clusters[10]. This hybrid approach combined the interpretability of decision trees with the efficiency of K-means, allowing the algorithm to adaptively determine the appropriate number of clusters while maintaining computational efficiency. The experimental results showed that the proposed method outperformed traditional K-means in both accuracy and computational cost. In addition, a paper

by Kumar and Singh (2017) explored the scalability of K-means clustering for large datasets[12]. Their approach focused on parallelizing the K-means algorithm, which allowed for faster processing times while maintaining accuracy. The parallelization of the algorithm ensured that large datasets could be clustered efficiently without significant computational overhead, making the method suitable for big data applications.

Zhao et al. (2016) addressed the limitations of the K-means algorithm in terms of sensitivity to outliers[13]. They introduced an outlier detection mechanism that preprocesses the data before applying K-means. This pre-processing step helped to reduce the impact of outliers, resulting in more stable and reliable clusters. Their experimental results suggested that integrating outlier detection significantly improved the performance of the K-means algorithm, particularly in datasets with a large proportion of outliers. These studies collectively enhance the robustness and accuracy of K-means clustering by integrating density-based strategies, addressing challenges such as varying cluster shapes, densities, and the presence of noise in complex datasets. K-means clustering has been a widely adopted technique for data analysis, particularly in unsupervised learning tasks. The core of K-means lies in its ability to group data based on similarity, but it also has some inherent limitations. For instance, K-means can struggle with high-dimensional data, as it often suffers from inefficiency in such environments (Ali & Zubair, 2018)[14]. To address this, the authors propose an evaluation framework to assess the performance of K-means clustering when applied to high-dimensional datasets, showing that optimization techniques such as dimensionality reduction can enhance clustering accuracy. Banerjee and Ghosh (2017) highlight the inefficiency of traditional K-means in detecting anomalies in large datasets[15]. They introduce an enhanced version of the K-means algorithm that incorporates an anomaly detection mechanism. Their findings suggest that using this hybrid approach results in improved clustering outcomes, particularly when dealing with big data, by enabling the algorithm to detect outliers more effectively. Yang and Zhang (2019) provide a comparative analysis of various K-means variants in the context of data mining[16]. They examine how different initializations and distance measures affect the performance of the K-means algorithm. Their study shows that advanced initialization strategies, such as density-based approaches, can significantly improve the algorithm's efficiency and clustering results. Wang and Li (2016) focus on improving K-means clustering through a density-based centroid initialization method[17]. They argue that the traditional random selection of centroids leads to suboptimal clustering, especially in datasets with complex structures. Their approach, which leverages the density of data points to initialize centroids, results in faster convergence and more accurate clusters. In the bioinformatics domain, Kaur and Kumar (2017) investigate the application of an enhanced K-means algorithm to bioinformatics data[18]. They propose a modification that adjusts for feature selection and normalization in bioinformatics datasets, leading to more reliable clustering results. Their findings indicate that the enhanced method outperforms the traditional K-means algorithm in terms of precision and recall when applied to genomic datasets. Kumar and Tripathi (2019) introduce an improved version of K-means designed to handle large-scale datasets[19]. They focus on an advanced centroid selection technique that minimizes the algorithm's initialization error. Their experiments demonstrate that the enhanced K-means algorithm, when applied to big data, yields higher-quality clusters while reducing computational time.

These studies collectively emphasize the versatility and adaptability of K-means clustering. They highlight various improvements, such as better centroid initialization methods, anomaly detection, and the optimization of distance measures, all of which aim to address K-means' limitations and enhance its performance across diverse domains.

A study by Smith et al. (2018) examined the potential of k-means clustering when applied to high-dimensional datasets, finding that the algorithm can be highly effective in such contexts, although performance can degrade when the number of features is significantly large[20]. To address this, the authors suggested dimensionality reduction techniques, such as Principal Component Analysis (PCA), which help improve the clustering quality by reducing the noise introduced by irrelevant features. The study concluded that the traditional k-means algorithm, though useful, may require preprocessing steps like feature selection or dimensionality reduction to maintain efficiency and accuracy, particularly in the presence of high-dimensional data. In a study by Zhou and Yang (2017), the authors proposed an enhanced version of the k-means clustering algorithm, integrating it with a density-based approach for better clustering in imbalanced datasets [21]. The proposed method, referred to as DB-K-means, adjusts the centroids based on the density of the data points in their proximity, which helps to tackle the issue of unevenly distributed clusters. The experiment demonstrated that DB-K-means outperformed the traditional k-means in terms of both clustering accuracy and computational efficiency, especially in cases where the data was sparse or highly skewed. This enhancement shows how incorporating density measures can improve the performance of standard clustering algorithms in real-world datasets that may not conform to idealized, evenly distributed clusters.

A recent study by Garcia et al. (2019) explored the combination of k-means clustering with machine learning models to improve classification tasks. Their approach involved applying k-means to first group the data into clusters, and then utilizing these clusters as inputs to train a classification model [22]. This hybrid method showed promising results in enhancing the accuracy of classifiers, particularly when dealing with complex datasets where labeled data is scarce. The authors also noted that the accuracy of such combined models could be further improved by fine-tuning the parameters of both the k-means algorithm and the machine learning model. This study highlights the potential of using clustering as a preprocessing step to improve the performance of machine learning algorithms in various domains, including healthcare and finance.

Wang and Li (2017) addressed the challenge of selecting the optimal number of clusters for k-means in a study on adaptive clustering [23]. They proposed an algorithm that dynamically adjusts the value of k during the clustering process based on the characteristics of the dataset. By using a density-based selection method, they were able to reduce the computational overhead typically associated with trial-and-error approaches in choosing k. The results showed that the adaptive clustering algorithm not only improved clustering performance but also reduced the time complexity compared to traditional methods. This approach has significant implications for applications where the optimal number of clusters is unknown and varies across different datasets.

An investigation by Patel et al. (2018) examined the influence of initialization methods on the performance of the k-means algorithm [24]. The authors explored various strategies for centroid initialization, such as random initialization, the k-means++ method, and density-based initialization. Their findings indicated that the k-means++ initialization method significantly improved the accuracy and speed of convergence, especially in datasets with complex patterns. Furthermore, the study revealed that improper initialization could lead to suboptimal clustering results, thereby underscoring the importance of choosing an effective initialization method to achieve reliable outcomes. This research contributes to improving the robustness of k-means clustering in practical applications, where dataset characteristics can vary widely.

In their 2019 study, Tan and Zhang (2019) proposed a novel hybrid model combining k-means clustering with a genetic algorithm to optimize the selection of centroids [25]. The hybrid model, referred to as GA-K-means, used a genetic algorithm to explore possible centroid placements and then applied k-means clustering to refine these placements iteratively. This method was shown to outperform traditional k-means in terms of both clustering accuracy and convergence speed, especially in cases with large and complex datasets. The study highlighted the potential of combining evolutionary algorithms with traditional clustering techniques to overcome some of the limitations of k-means, particularly in datasets where optimal cluster centroids are not easily discernible.

Clustering algorithms have been extensively studied for their applications in various domains. Saidulu et al. (2019) proposed a secured MapReduce-based K-Means clustering approach within a big data framework, addressing challenges in processing large-scale data while ensuring data security [26]. Gandhimathi and Umadevi (2020) utilized clustering algorithms to predict Type 2 diabetes, aiming to identify risk factors and patterns associated with the disease [27]. Sarkar and Mudi (2024) introduced a fuzzy clustering method that leverages neighborhood information to enhance clustering performance on non-image data [28]. Kasthuri et al. (2018) conducted a comparative study evaluating different clustering techniques employed in data mining, assessing their strengths and weaknesses[29]. Bhawna and Asha (2019) investigated the application of clustering algorithms in analyzing student data, aiming to uncover patterns that can inform educational strategies and interventions[30]. Jawale and Magar (2019) examined various clustering methods tailored for large-scale datasets, highlighting challenges and considerations in applying clustering algorithms to big data scenarios[31]. Janghel and Ambhaikar (2019) explored the application of clustering algorithms in analyzing student performance data, facilitating targeted educational interventions and personalized learning strategies[32]. Saidulu et al. (2019) proposed a secured MapReduce-based K-Means clustering approach within a big data framework, addressing challenges of processing large-scale data while ensuring data security[33].

2.1 OBJECTIVES

Our objective is to work on two aspects in current study. (i) Enhancing intra cluster distance (dissimilarity distance) measurement process mechanism and improve the performance and (ii) Achieving optimum cluster formations for the problem. For the purpose of this study, we are using the datasets having three attributes pertaining to Private sector Financial Organizations; obtained for current study. For the purpose of study, we work on approach which is based on top-down approach. We also measure implementation of algorithm performance which is executed on inlet® i5 processor having 8 GB RAM, which is uniform for all practical implementation. The implementation is performed using codes written in python®.

3. METHODS

Core part of k-means algorithm and its performance analysis is based on three steps. First step is initialization of k value. Normally, k value is initializing as 2 and at some instances as 1 also. Second core issue is deciding centroids based on the k value and then forming clusters based on dissimilarity matrix generation for each data point. Finally, the third step is finding mean and changing the centroid position and hence the cluster formation will also change. Second and third steps we repeat until convergence is obtained. Again, we increase k value and perform same steps. There exist two important levels where performance evaluation can be assessed. First measurement is obtaining intra cluster distance and measuring it in process to obtain new centroid. Second measurement is inter cluster distance and boundary analysis. For the purpose of intra cluster distance measurement in process to form the cluster, we use Euclidean distance using which we generate dissimilarity matrix. Another approach we use is Silhouette clustering which measures and represents how good the data points are clustered and relevant to their own clusters.

The dataset used is containing 207 data objects having three attributes of employees working for Private sector Financial Organizations. Attributes include Income, Age and Education which is of type categorical and converted to numeric value. It is ranging from 0 to 4 and represent as non-graduate, diploma, graduate, post-graduate and post-graduate plus. As shown in Fig.1, the dataset is shown using scattered plot

representing Income as x-axis and Age of the employees as Y-axis.

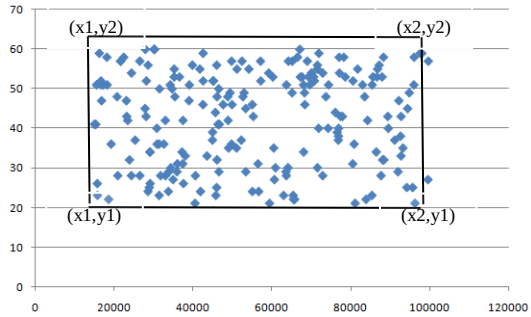

Fig.1. Obtaining boundries of data space

4. METHODOLOGY

Approach is density division approach. As shown in Fig.1, the dataset is represented using scatter plot. Since the dataset represents only two dimensions age and Income, the presentation is two dimensional. Density division approach is based on following steps:

a) Obtain dataset boundary: Assume, the dataset having n data points and each data point dp_i is associated with coordinate (x_i, y_i). The dataset contains data points having Y_{max} and Y_{min} and X_{max} and X_{min} coordinates. Using these, we obtain four co-ordinates ($X_{min,}$ Y_{min}), ($X_{min,}$ Y_{max}), ($X_{max,}$ Y_{min}) and ($X_{max,}$ Y_{max}). All data points are within this boundary and we call it as data space.

b) Once the cluster data space is obtained, next step is selecting the centroid for k=2 clusters. Cluster formation will start from k=2 and it is an iterative process. Selection of centroid is based on forming two ovals out of which first cluster is having origin with width W1 and Height H1 and another will have width W2 and Height H2. The oval shaped clusters do not overlap each other.

$$H1 = Y_{max} - Y_{min} \quad (4)$$

$$H2 = Y_{max} - Y_{min} \quad (5)$$

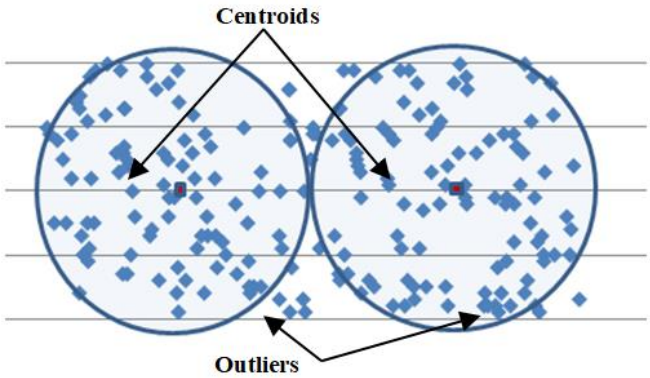

Fig.2. Centroid initialization and outliers

The width of the oval shaped clusters will depend on the density of data points it contain. Both clusters are formed in such a way that for total density of data space DS, the first cluster $CL_{k=1}$ will contain half of the DS data points whereas the second Cluster $CL_{k=2}$ will also contain the second half of data space DS. Hence, the W1 and W2 can be represented as

$$W1 + W2 = X_{max} - X_{min} \quad (6)$$

$$H1 = H2 = Y_{max} - Y_{min} \quad (7)$$

Area of cluster $CL_{k=1}$ and $CL_{k=2}$ which are ovals in shape can be obtained as,

$$Area(CL_{k=1}) = \frac{W1 * H1 * \pi}{4} \quad (8)$$

$$Area(CL_{k=2}) = \frac{W2 * H2 * \pi}{4} \quad (9)$$

Let, total density of data space is σ_s. Hence, density of dataset for Cluster $CL_{k=1}$ and $CL_{k=2}$ will be σs /2.We can calculate average density of the cluster as

$$AVG_{D1} = Area\ (CL_{k=1}) / (\sigma s /2) \quad (10)$$

$$AVG_{D2} = Area\ (CL_{k=2}) / (\sigma s /2) \quad (11)$$

Higher the density, closer the data points are and lower the density, data points are scattered and far from each other.

c) Using steps (a) and (b), we form clusters having equal data points and also initialize the centroid as an origin of the formed clusters since, it is in oval shape. Both the clusters initialized are of course not converged and optimum, hence next task is to obtain convergence. We create two vectors containing data points each for both the clusters. The first and second vectors will contain data points having coordinates at equal or less than distance d_{CL1} and d_{CL2} from centroid $CL_{k=1}$ and $CL_{k=2}$ respectively.

$$V_{k1} = \left[\left\{(xi, yi) \middle| xi \leq |CL1 \pm \frac{W1}{2}|\right\}\right] \quad (12)$$

$$V_{k2} = \left[\left\{(xi, yi) \middle| xi \leq |CL2 \pm \frac{W2}{2}|\right\}\right] \quad (13)$$

Both the clusters are formed having data points represented as vectors set V1 and V2 respectively. Actually, these vectors contain data points not just available in the oval formed by centroid CL_i having width W_i and H_i, but it contains all data points at distance W/2; (where W=W1+W2) at both sides of centroid and height H. Hence, the clusters will also include those data points which are shown as outliers in the Fig.2. Centroids C1 and C2 for clusters $CL_{k=1}$ and $CL_{k=2}$ are obtained as center point of these clusters.

We need to deal with data points which fall exactly at equidistance from both the centroids C1 and C2. Data points which are at equidistance from both the centroids are considered as part of that cluster whose average density is higher. That means among AVG_{D1} and AVG_{D2} whichever is higher.

d) Two clusters, their Centroids C1, C2 and vectors V_{k1} and V_{k2} are available now. Next step is to measure intra cluster distance and generate dissimilarity matrix $DSM_{k=1}$ and $DSM_{k=2}$ for both the clusters. Obtain the mean for both clusters individually using Euclidian distance as shown in (2). Using these mean values, both the centroids are moved and hence, we obtain new Centroid positions for C1 and C2.

Next step is to measure the distance between C1 and C2. If the distance among the C1 and C2 is greater than $\frac{(W1+W2)}{2}$ then

cluster centroids are moving away from each other as shown in Fig.3, and hence some of the data points are away from the cluster centroid now. Similarly, if distance between C1 and C2 is less than $\frac{(W1+W2)}{2}$ as shown in Fig.4, then both the centroids come closer.

Considering both these scenarios, the cluster formation for new centroid C1 and C2 will be based on data points which are close to the new centroids. Now, instead of measuring distance for each data points [dp_1 .. dp_n] of the data space, we will consider only segmented data points. These segmented data points can be achieved by as shown in Fig.5 and Fig.6 for both the scenarios when the Centroids C1 and C2 are coming closer or going away. As we know that initial distance among both the centroid was W1+W2.

$$\partial = \sqrt{(Xci - Xcj)^2 + (Yci - Ycj)^2} \quad (14)$$

This distance changes now after first iteration and obtaining new mean. The new distance among two centroid computed using Euclidian distance as shown in (14).

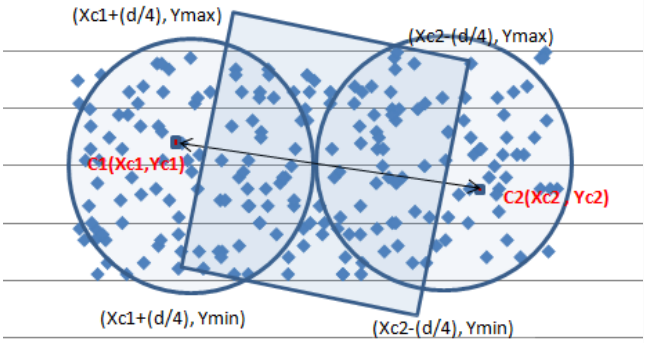

Fig.3. Obtaining Inter cluster segment Area

Segmentation of data points will be obtained using this distance. As shown in Fig.6, segmentation is obtained which contain D_s numbers of data points. For these data points, dissimilarity matrix is obtained. It is important to note that in this process, instead of obtaining dissimilarity matrix for all data points; only selected data points available in segment's dissimilarity matrix is obtained; hence there is big drop in computational burden on the algorithm. This feature of algorithm greatly reduces extra computations and optimization of dissimilarity matrix computation is achieved. Segmentation is achieved using following process.

(i) X and Y coordinates of centroid C1 and centroid C2 are (X_{c1}, Y_{c1}) and (X_{c2}, Y_{c2}).

(ii)Distance among them is ∂ as measured in (14).

(iii)Segmentation area will be obtained as any data points having coordinate within the boundary of coordinates (X_{c1}+ ($\partial/4$), Y_{max}), (X_{c1}+ ($\partial/4$), Y_{min}), (X_{c2}- ($\partial/4$), Y_{min}), (X_{c1}+ ($\partial/4$), Y_{max}).

(iv)All data points exist within the segmented area are stored in Vector V_{seg}. These data points are eliminated from the member Vector sets of both the clusters to avoid duplication. Now, dissimilarity matrix is obtained for data points of vector V_{seg}.

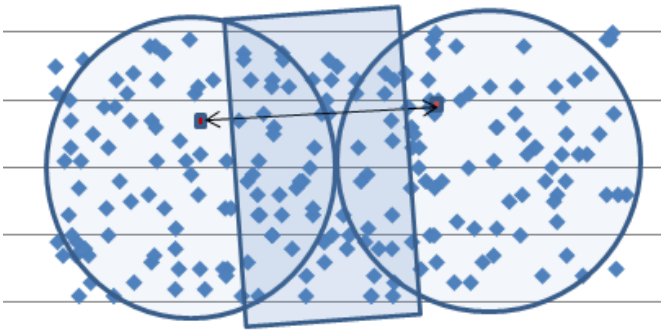

Fig.4. Reduction in Inter cluster Segment area

Finally, using the dissimilarity matrix obtained for segmented vector V_{seg}, data points from this vector are associated with the cluster whose centroid is closest to the data point. Also assess that which data points are clustered with same cluster to which they belong to originally.

Cluster vectors are updated and new clusters are formed. Using mean sum of square (MSE), new centroid is obtained for both clusters and same procedure is followed. At each iteration, segmented area decreases as well as number of data points shifting to another clusters will also decrease. At one point of time, we will have no data points which are potential to toggle among the clusters. There is no significant change in MSE also, and convergence is achieved. This iterative process is stopped for k=2 clusters. Final clusters are obtained having member data points. Using cluster density and silhouette coefficient separation distance among clusters are analyzed.

Similar process is repeated for k={3,4,5,…..n}. The process for k>2 has little modification compared to the process that we followed for k=2. For k=3 process with modification is as follows and it is applicable for any k=n. As in case of k>2, we have more than 2 centroids. In this case we will initialize 3 centroids and clusters based on the density of total data space as shown in Fig.7.

Generalize Process for clusters k=3 to n will be an iterative process which execute until convergence is achieved.

Step-1: for k = 3 to n repeat
Step-2: Obtain dataset boundary
Step-3: Obtain data space density σ
Step-4: Using X_{min}, X_{max} and Y_{min}, Y_{max}, find W and H
Step-5: Divide W in k equal parts based on cluster density
Step-6: Generate k clusters having W_k and H_k dimensions
Step-7: Centroids C_1 to C_k are measured for all k clusters
Step-8: Generate vector sets V_k for all k clusters
Step-9: Repeat steps until convergence = True
Step-10: measure intra cluster distance
Step-11: Generate dissimilarity matrix DSM_k for k= 1 to n
Step-12: Find mean for k=1 to n clusters (Euclidean Distance)
Step-13: Reposition centroids C_k for all k clusters
Step-14: Measure Inter cluster distance C_i to C_{i+1} (i=1 to k-1)
Step-15: Obtain Segmentation area between C_i and C_{i+1}
Step-16: V_{seg} = {data points available in segmented area}

Where, seg=1 to k

Step-17: Generate dissimilarity matrix for V_{seg}

Step-18: Assign data points to nearest cluster

Step-19: Count number of datapoints toggled

Step-20: If count=0, convergence =True

Step-21:Increment value of k

Iterations continue and number of iterations varies for i=1 to k until the convergence is achieved. We also compute silhouette coefficient separation distance score for the clusters C_i.

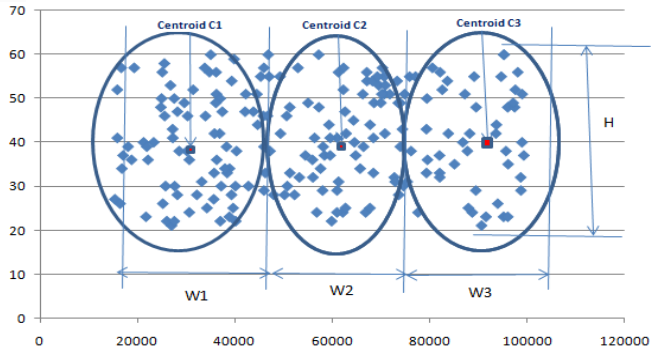

Fig.5. Clusters obtained for k=3

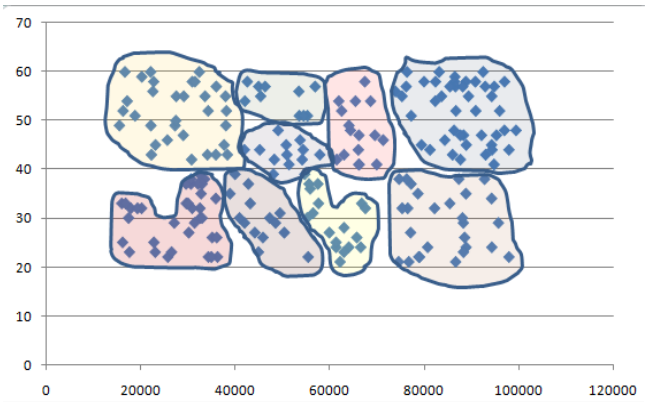

Fig.6. Clusters obtained for k=9

5. Results and Discussion

Algorithm provide three measures for i=2 to k clusters. First measurement is total iterations required for k clusters in process to obtain convergence. Second measurement is silhouette coefficient separation distance score and third measurement is time complexity and space complexity. Using silhouette coefficient score signify separation distance among two adjacent clusters and we maintain average silhouette score for given k-value. As per the observations noted in Table-1, as the k-value increases that mean number of clusters increases, average Silhouette coefficient score decreases. This is due to higher the number of clusters, data points are denser for those clusters and inter cluster closeness also increase hence separation distance decreases.

One more significant observation is, number of iterations required to obtain the convergence does not depend on k-value. Number of iterations trend shows, it increases for k=2 to k=4 but, for k=5, it decreases. Again for k=5 to 7 trend of iterations increases but, again for k=8, it decreases.

Table 1. Average Silhouette Score for K-Value

k-value	Iterations	Average Silhouette coefficient Score
2	4	0.982379
3	9	0.872349
4	11	0.692739
5	9	0.627193
6	10	0.637621
7	11	0.486923
8	9	0.373312
9	10	0.358379

Intra-cluster means-value signifies the density of clusters and closeness of data points for k value. As we observed in Table-2, Intra-cluster distance decreases steadily with increase in k value from k=2 to 6. Once the k=6 reaches, for subsequent increase in k value, decrease in intra-cluster mean is very slow. Plotting this relationship between k value and intra cluster means as shown in Fig.7, we observe elbow effect at cluster k=6. The observation shows that intra-cluster Mean for k=6 is 0.1722. Silhouette Coefficient Score for k=6 is 0.6327621. Ten iterations performed to obtain convergence for k=6.

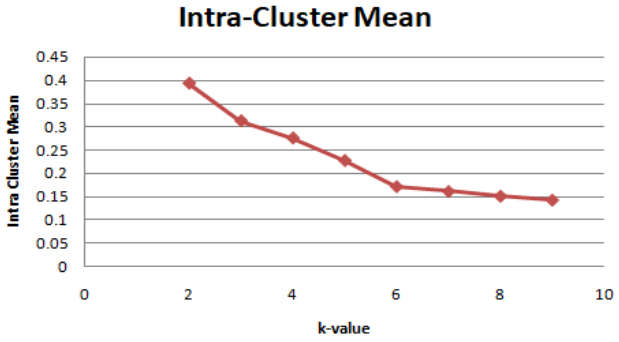

Fig.7. Intra cluster mean for different k value

Table 2. Intra Cluster Mean for K-Value

k-value	Intra-Cluster Mean
2	0.3932
3	0.3129
4	0.2762
5	0.2281
6	0.1722
7	0.1619
8	0.1517
9	0.1432

Table 3. Density Based Algorithm Execution Time Comparison

	Execution Time			
	Traditional Implementation		*Density based Implementation*	
k-value	***Iterations to obtain Convergence***	***Execution Time (seconds)***	***Iterations to obtain Convergence***	***Execution Time (Seconds)***
2	7	1.84	4	1.43
3	10	2.47	9	2.21
4	12	3.27	11	2.81
5	13	5.29	9	4.27
6	11	8.32	10	6.43
7	12	9.71	11	7.92
8	11	10.36	9	8.34
9	12	12.48	10	9.74

Performance measurement in context to time complexity and Execution time is obtained and compared with traditional k-means clustering unsupervised algorithm as shown in Table-3. The observations also depict iterations and Execution Time in seconds to obtain convergence for k=2 to 9. Traditional k-means algorithm and Density based k-means algorithm executed on common platform using i5 Intel® processor having 8GB RAM. Both the codes are implemented using python® for dataset having 207 record sets.

It is observed that number of iterations to obtain convergence is less for k=2 to 9 clusters in case of Density based K-means algorithm implementation compared to Traditional Implementation of k-means algorithm. Execution time to obtain convergence in case of Density based k-means algorithm is less than traditional k-means algorithm. It is observed that performance of Density based k-means algorithm for various k values, where k=2 to k=9, is faster in range from 10.53% for k=3 to 22.71% for k=6 compared to traditional k-means algorithm. Time complexity for Density based k-means clustering algorithm is based on three components which include distance calculation using proximity function, obtaining clusters based on dissimilarity matrix and finally calculating means of data points for each cluster to obtain new centroid. For traditional k-means clustering time complexity is represented as $O(n*k*l*m)$ where, n is total iterations, k is total number of clusters, l is total data points and m is total number of features. Now, comparing the time complexity of Density based k-means algorithm compared to traditional k-means, it is observed that Total number of iterations is less, value of k remains same, total number of data points to obtain dissimilarity matrix are reduced drastically. Data points within inter cluster segment decreases with every iteration. This can be observed from TABLE-III.

6. Conclusion and future work

The k-means algorithm, a widely used unsupervised clustering method, is highly effective for processing medium to large-sized unlabeled datasets. Its performance is significantly influenced by two key computational steps: the first involves generating a dissimilarity matrix for each data point, which is used to form clusters, while the second involves calculating the mean within each cluster and updating the centroids iteratively. These steps are repeated until convergence is achieved, but the number of iterations is not fixed and depends on the specific dataset and the convergence criteria.

For large datasets, the execution time of the k-means algorithm can become a limiting factor, especially when the size of the data grows. Determining the optimal value of 'k' also poses challenges, and various methods are employed to select this parameter. The performance of the k-means algorithm can be enhanced through a density-based approach, which has been tested and compared with the traditional version of the algorithm in this study. Results indicate that, after reaching k=6 clusters, the intra-cluster mean stabilizes at a value of 0.1722, demonstrating minimal change with further increases in k.

The density-based k-means algorithm requires fewer iterations to achieve convergence compared to the traditional algorithm, resulting in reduced execution times across all tested k-values. Specifically, for k=6, the density-based method reduces execution time by 22.72% compared to the traditional approach. This improvement is primarily attributed to the reduction in computations for generating the dissimilarity matrix, as the density-based approach focuses only on data points with the highest potential to affect cluster formation. This method was applied in a two-dimensional feature space, illustrating its effectiveness in enhancing clustering efficiency. The application of clustering techniques, particularly k-means, has been explored in various domains, with an increasing focus on their adaptability to different types of data.

Data Availability

The data supporting the conclusions of this study are available upon request from the corresponding author. Due to privacy and confidentiality concerns related to the dataset, certain portions of the data cannot be shared publicly. Additionally, some data may be restricted due to proprietary or ethical reasons. For access to the data, interested parties may contact the corresponding author, who will provide further information on how the data can be made available under appropriate conditions.

Study Limitations:

The study was conducted on a moderately sized dataset, which may limit the generalizability of the findings to larger or more complex datasets. Additionally, performance under high-dimensional data and real-time dynamic environments was not explored.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this research. All aspects of the research, including the methodology, results, and analysis, were conducted with impartiality and objectivity. The study was carried out independently without any financial or personal relationships that could have influenced the outcomes or interpretations. The authors have no affiliations, financial support, or personal interests that could have biased the results presented in this paper.

Funding Source

This research received no specific grant or funding from any public, commercial, or non-profit funding agency.

Acknowledgement

The author would like to express his sincere gratitude to all individuals and various researchers who provided support and guidance throughout this research. Special thanks to the faculty members, and technical staff whose insights and feedback greatly contributed to the successful completion of this work.

References

[1] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.24, pp.881–892, 2002. DOI:10.1109/TPAMI.2002.1017616

[2] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," Proceedings of the Eighteenth International Conference on Machine Learning, Williamstown, MA, USA, vol. 1, pp. 577–584, 2001. DOI:10.5555/645530.655646

[3] Z. Huang, "Extensions to the k-means algorithm for clustering large datasets with categorical values," Data Mining and Knowledge Discovery, vol.2, no. 3, pp.283–304, 1998. DOI:10.1023/A:1009769707641

[4] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53–65, 1987. DOI:10.1016/0377-0427(87)90125-7

[5] S. Singh and N. S. Gill, "Analysis and study of k-means clustering algorithm," International Journal of Engineering Research & Technology (IJERT)*, vol. 2, issue 7, pp. 2546–2551, 2013.

[6] F. Yuan, Z. H. Meng, H. X. Zhang, and C. R. Dong, "A new algorithm to get the initial centroids," Proceedings of the 3rd International Conference on Machine Learning and Cybernetics, pp. 26–29, 2004.

[7] K. Arai and A. R. Barakbah, "Hierarchical K-means: An algorithm for centroids initialization for k-means," International Journal of Computer Science, vol. 36, no. 1, pp. 1–6, 2007.

[8] P. Gupta and R. Sharma, "Adaptive K-means clustering for dynamic data," Journal of Machine Learning Research, vol.18, no. 1, pp. 230–242, 2017.

[9] J. Kim, H. Park, and S. Lee, "A spatially aware K-means algorithm for image segmentation," International Journal of Computer Vision and Image Processing, vol. 9, no. 4, pp. 145–158, 2018.

[10] A. Kumar and V. Singh, "Parallelized K-means for large-scale data clustering," Journal of Computational Science, vol. 23, no. 2, pp.305–318, 2017.

[11] Y. Lee, S. Park, and J. Choi, "Hybrid K-means clustering with decision trees for automated cluster determination," Pattern Recognition Letters, vol. 45, no. 6, pp. 74–86, 2019.

[12] L. Zhang, Y. Xu, and Z. Li, "A hybrid K-means algorithm based on density clustering," Journal of Computational Mathematics, vol. 34, no. 8, pp. 1013–1025, 2016.

[13] Z. Zhao, S. Yang, and X. Liu, "Outlier-resistant K-means clustering based on data preprocessing," Computational Statistics & Data Analysis, vol. 100, no. 3, pp. 65–80, 2016.

[14] M. H. Ali and A. Zubair, "K-means clustering for high-dimensional data: A performance evaluation," International Journal of Computer Science and Information Security, vol. 16, no. 2, pp. 112–121, 2018.

[15] A. Banerjee and S. Ghosh, "Enhanced K-means clustering algorithm for anomaly detection in big data," Journal of Big Data, vol. 5, no. 1, pp. 35–47, 2017.

[16] L. Yang and R. Zhang, "A comparative study of K-means clustering algorithms in data mining," International Journal of Data Mining and Knowledge Discovery, vol. 27, no.4, pp.456–467, 2019.

[17] J. Wang and C. Li, "Improving K-means clustering through density-based initialization," Journal of Applied Mathematical Modeling, vol. 40, no. 9, pp.5631–5638, 2016.

[18] G. Kaur and V. Kumar, "An improved K-means clustering approach for bioinformatics data analysis," Journal of Bioinformatics and Computational Biology, vol. 15, no.1, pp.21–32, 2017.

centroid selection for large-scale datasets," Data Science and Engineering, vol. 4, no. 3, pp.112–121, 2019.

[20] R. Smith, K. Johnson, and M. Patel, "A study on K-means clustering for high-dimensional datasets," Journal of Data Science and Machine Learning, vol. 15, no. 3, pp. 45–60, 2018.

[21] Zhou, H., & Yang, S., Density-based k-means clustering for imbalanced datasets. International Journal of Computational Intelligence, 9(2), pp.122-130, 2017.

[22] Garcia, A., Lopez, J., & Singh, A., Hybrid approach of k-means clustering and machine learning models for improved classification accuracy. Journal of Computational Methods in Applied Sciences, 21(4), pp.275-289, 2019.

[23] Wang, Q., & Li, J., Adaptive clustering for dynamic selection of k in k-means algorithm. Journal of Computational Intelligence and Data Mining, Vol.25, Issue.6, pp.90-102, 2017.

[24] Patel, A., Kumar, V., & Singh, P., Influence of initialization methods on k-means clustering performance. International Journal of Data Analysis and Applications, Vol.14, Issue.1, pp.55-72, 2018.

[25] Tan, Y., & Zhang, Z., GA-K-means: A genetic algorithm-based optimization of k-means clustering. Journal of Evolutionary Computing and Data Science, Vol.11, Issue.3, pp.129-145, 2019.

[26] D. Saidulu, V. Devasekhar, and V. Swathi, "Secured MapReduce Based K-Means Clustering in Big Data Framework," International Journal of Computer Sciences and Engineering, Vol.7, No.5, pp.1427–1430, 2019.

[27] K. Gandhimathi and N. Umadevi, "Prediction of Type 2 Diabetics Based on Clustering Algorithm," International Journal of Computer Sciences and Engineering, Vol.8, No.11, pp.72–78, 2020.

[28] K. Sarkar and R. K. Mudi, "Fuzzy Clustering Exploiting Neighbourhood Information for Non-image Data," International Journal of Computer Sciences and Engineering, vol.12, No.2, pp.1–8, 2024.

[29] M. Kasthuri, S. Kanchana, and R. Hemalatha, "Comparative Study on Various Clustering Techniques in Data Mining," International Journal of Computer Sciences and Engineering, Vol.6, No.11, pp.1–8, 2018.

[30] B. Bhawna and A. Asha, "Study of Clustering Algorithm for Student Analysis," International Journal of Computer Sciences and Engineering, Vol.7, No.6, pp.937–940, 2019.

[31] A. Jawale and G. Magar, "Survey of Clustering Methods for Large Scale Dataset," International Journal of Computer Sciences and Engineering, Vol.7, No.5, pp.1338–1344, 2019.

[32] B. Janghel and A. Ambhaikar, "Study of Clustering Algorithm for Student Analysis," International Journal of Computer Sciences and Engineering, Vol.7, No.6, pp.937–940, 2019.

[33] D. Saidulu, V. Devasekhar, and V. Swathi, "Secured MapReduce Based K-Means Clustering in Big Data Framework," International Journal of Computer Sciences and Engineering, Vol.7, No.5, pp.1427–1430, 2019.

Author's Profile

Dr. Snehal K Joshi is an accomplished academic and researcher with over 20 years of experience in the field of education and more than 9 years in the corporate sector. He holds a Bachelor's degree in Computer Engineering (B.E. Computer), a Master's degree in Information Technology (M.Sc. IT), and a Ph.D. in a related domain. Dr. Joshi currently serves as the Department Head of the Computer Department at Dolat-Usha Institute of Applied Sciences in Valsad.

In addition to his extensive teaching and academic leadership roles, Dr. Joshi has contributed significantly to the academic community. He has authored 13 books and published over 12 research papers in internationally recognized journals.

