Implementation of K-means, K-medians, K-medoids and Bisecting K-means clustering algorithms and comparing their performances

Somsubhra Mukherjee(CST-510518004)

Mohammad Naimuddin Momin(CST-510518007)

Ayush Nag(CST-510518011)

Subhendu Poddar(CST-510518037)

Under the mentorship of **Professor Somnath Pal**

Department of Computer Science and Technology

Indian Institute of Engineering Science and Technology, Shibpur

CONTENTS

TOPIC	PAGE NO.
1. INTRODUCTION	01
2. PARTITIONED CLUSTERING ALGORITHMS	02
3. HIERARCHICAL CLUSTERING ALGORITHMS	07
4. VALIDITY INDICES	10
5. EMPIRICAL EVALUATION	21
6. CONCLUSION	40

7. REFERENCES 41

8. 1. INTRODUCTION

The method of identifying similar groups of data, which are unlabelled, in a dataset is called clustering. It is one of the most important techniques in data science. Entities in each group are comparatively more similar to entities of that group than those of the other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

The purpose of clustering algorithms is to make sense of and extract value from large sets of structured and unstructured data. If we're working with huge volumes of unstructured data, it only makes sense to try to partition the data into some sort of logical groupings before attempting to analyze it. Clustering allows us to take a sweeping glance of our data en masse, and then form some logical structures based on what we find there before going deeper into the analysis. It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

With the development of a number of clustering algorithms, the need to compare their performances and to determine which clustering algorithm yielded the best clustering, also rose. To evaluate the quality of partitioning, some statistics called validity indices were proposed. They assumed as much importance as the clustering algorithms themselves in the context of data mining as they are used to determine the algorithm giving the best results.

We have implemented three partitioned clustering algorithms, namely K-means, K-medians and K-medoids clustering algorithms and a hierarchical clustering algorithm, namely Bisecting K-means clustering algorithm. The next two sections describes these types of algorithms. Section 4 contains detailed discussion on four very popular validity indices – Dunn index, Davies-Bouldin(DB) index, Silhouette index(SI), and CDbw index. Section 5 reveals the experimental design and the results obtained using 15 different real-life datasets. Section 6 summarizes our conclusion and section 7 includes a few references.

2. PARTITIONED CLUSTERING ALGORITHMS

Partitioned clustering decomposes a data set into a set of disjoint clusters. Given a data set of N points, a partitioning method constructs K ($K \leq N$) partitions of the data, with each partition representing a cluster. That is, it classifies the data into K groups by satisfying the following requirements: (1) each group contains at least one point, and (2) each point belongs to exactly one group. Many partitioned clustering algorithms try to minimize an objective function by partitioning the data.

In this project, we have worked on 3 partitioned clustering algorithms, namely K-means, K-medians and K-medoids clustering which have been discussed in the following 3 subsections.

2.1: K-MEANS CLUSTERING ALGORITHM

K-means **[1]** algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It assigns data points to a cluster such that the sum of the Euclidean distances (L2 norm) between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is the minimum. The approach K-means follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster.

```
ALGORITHM FOR K-MEANS CLUSTERING

1: Specify the number of clusters K.
2: Initialize centroids by randomly selecting K points for the centroids
   without replacement.
3: repeat
4:     Compute the sum of the squares distances (Euclidean distances)
       between data points and all centroids.
5:     Assign each data point to the closest cluster (centroid).
6:     Re-compute the centroids for the clusters by taking the average(mean)
       of the all data points that belong to each cluster.
7:     Calculate the sum of square errors (SSE) by the following formula:

       $SSE = \sum_{1 \le k \le K} \sum_{xij \in Ck} |x_{ij} - c_{kj}|^2$

8: until (The newly calculated centroids does not change with respect to
          previously calculated ones).
```

Figure 1: Algorithm for K-means clustering

LIMITATIONS OF K-MEANS ALGORITHM:

1. The number of clusters required should be specified apriori.
2. Given K-means iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since K-means algorithm may get stuck in a local optimum and may not converge to global optimum.

3.

2.2: K-MEDIANS CLUSTERING ALGORITHM

K-median clustering algorithm **[2]** is a variation of K-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median. This has the effect of minimizing error over all clusters with respect to the 1-norm (Manhattan) distance metric, as opposed to the squared 2-norm distance metric (which K-means does). The proposed algorithm alternates between an expectation (E) and maximization (M) step, making this an Expectation–maximization algorithm. In the E step, all objects are assigned to their nearest median. In the M step, the medians are recomputed by using the median in each single dimension.

```
ALGORITHM FOR K-MEDIANS CLUSTERING

1: Specify the number of clusters K.
2: Initialize centroids by randomly selecting K points for the centroids
   without replacement.
3: repeat
4:     Compute the sum of the absolute values of the distances (Manhattan
       distances) between data points and all centroids.
5:     Assign each data point to the closest cluster (centroid).
6:     Re-compute the centroids for the clusters by first sorting them and
       then taking the middlemost element (median) of all data points that
       belong to each cluster for each attribute.
7:     Calculate the sum of errors (SE) by the following formula:

       $SE = \sum_{1 \leq k \leq K} \sum_{xij \in Ck} |x_{ij} - c_{kj}|$

8: until (The newly calculated centroids does not change with respect to
          previously calculated ones).
```

Figure 2: Algorithm for K-medians clustering

LIMITATIONS OF K-MEDIANS ALGORITHM:

1. The number of clusters required should be specified beforehand.
2. Given the iterative nature of K-medians clustering and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since K-medians algorithm may get stuck in a local optimum and may not converge to global optimum.

2.3: K-MEDOIDS CLUSTERING ALGORITHM

K-Medoids **[3]** (also called as Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum. The K-medoids algorithm is a clustering algorithm reminiscent to the K-means algorithm. Both the K-means and K-medoids algorithms are partitioned (breaking the dataset up into groups) and both attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the K-means algorithm, K-medoids chooses data points as centers (medoids or exemplars) and can be used with arbitrary distances (especially the L1-norm or Manhattan distance).

```
ALGORITHM FOR K-MEDOIDS CLUSTERING

1:  Specify the number of clusters K.
2:  Initialize the medoids by randomly selecting $K$ points from within the
    given dataset for the medoids without replacement.
3: repeat
4:  Compute the sum of the absolute values of the distances (Manhattan
    distances) between data points and all medoids.
5:  Associate each data point to its closest cluster medoid .
6:  Now calculate the cost in each cluster C by the following formula:

    cost = $\sum_{x_{ij} \in C} |x_{ij} - m_{Cij}|$, where $m_C$ is the medoid in cluster C.

7:  Now find the total cost, that is, summation of costs of all the clusters.
      COST = $\sum (\text{cost})_k$
8:  While the COST decreases:
    For each medoid m, and each non-medoid data point o:
        i>Swap m and o, associate each data point to the closest medoid
        and re-compute the COST .
        ii>If the total cost(COST) is more than that in the previous step,
        undo the swap.

9:  Calculate the sum of errors (SE) by the following formula:
    SE = $\sum_{1 \le k \le K} \sum_{x_{ij} \in C_k} |x_{ij} - m_{kj}|$

10: until (The newly calculated medoids does not change with respect to
        previously calculated ones).
```

Figure 3 : Algorithm for K-medoids clustering

LIMITATIONS OF K-MEDOIDS ALGORITHM:

1. The number of clusters required should be specified beforehand.

2. It may obtain different results for different runs on the same dataset because the first *k* medoids are chosen randomly.

3. One of the main disadvantage of K-Medoid algorithms is that it is not suitable for clustering non-spherical (arbitrary shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster center) - briefly, it uses compactness as clustering criteria instead of connectivity.

4. The time complexity of K-medoid is O(n^2), unlike K-means which has a time complexity of O(n).

3. HIERARCHICAL CLUSTERING ALGORITHMS

Hierarchical clustering, also known as hierarchical cluster analysis, works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a separate cluster. Then, it repeatedly executes the following two steps: (1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This iterative process continues until all the clusters are merged together. The main output of Hierarchical Clustering is a dendrogram, which graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view). There are two types of hierarchical clustering, Divisive and Agglomerative.

Agglomerative: This is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

Divisive: This is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In this project, we have implemented one divisive hierarchical clustering method, namely the Bisecting K-means algorithm, which is described as follows.

BISECTING K-MEANS CLUSTERING ALGORITHM

Bisecting K-means clustering **[4]** is a hybrid approach between Divisive Hierarchical Clustering (top down clustering) and K-means Clustering. Instead of partitioning the data set into k clusters in each iteration, Bisecting K-means algorithm splits the parent cluster C into two daughter clusters C_1 and C_2 at each bisecting step (by using K-means with k=2) until k clusters are obtained. The advantage of this method lies in the fact that it is more efficient when k is large. For the K-means algorithm, the computation involves every data point of the data set and k centroids. On the other hand, in each bisecting step of Bisecting K-means, only the data points of one cluster and two centroids are involved in the computation. Thus, the computation time is reduced. Besides, Bisecting K-means produces clusters of similar sizes, while K-means is known to produce clusters of widely different sizes.

ALGORITHM FOR BISECTING K-MEANS CLUSTERING

```
1:  Specify the number of clusters K.
2: *repeat*
3:  Choose the parent cluster to be split C.
4:  *repeat*
5:      Choose two centroids at random from C; $C_1$ and $C_2$, without replacement.
6:      Compute the sum of squares (Euclidean) distance of each point from the two centroids.
7:      Assign each point to the nearest cluster (centroid).
8:      Re-compute the centroids for the clusters by taking the average(mean) of the all data points that belong to each cluster.
9:      Calculate the sum of square errors (SSE) by the following formula:
        $SSE = \sum_{k=1,2} \sum_{xij \in Ck} |x_{ij} - C_{kj}|^2$
10: *until* (The newly calculated centroids does not change with respect to the previously calculated ones)
11: Evaluate SSE for each cluster using the following formula:
        $SSE_k = \sum_{xij \in Ck} |x_{ij} - C_{kj}|^2$
12: Choose the cluster with larger value of SSE and set it as the parent cluster.
13: *until* (K clusters have been obtained.)
```

Figure 4: Algorithm for Bisecting K-means clustering

LIMITATIONS OF BISECTING K-MEANS ALGORITHM:

1. The time complexity for the clustering can result in very long computation times, in comparison to K-means.
2. The algorithm can never undo any previous steps. For example, suppose the algorithm clusters 2 points and later finds that the connection was not a good one, it cannot redo the step.
3. Finally, if we have a large dataset, then it can become difficult to determine the correct number of clusters by dendogram.

4. VALIDITY INDICES

Together with specification of elementary clusters, it is necessary for data miners to specify the quality of such clusters. In spite of the diversity of clustering algorithms, their common aim is detection of clusters, which are the most compact and separable. The compactness indicates how close the objects in a cluster are. For example if we consider the variance of objects, the lower the value of the variance, the higher the compactness of the cluster. Compact clusters are the best ones, so low values of compactness are desired. Separability indicates how distinct the clusters are. One way to express separability is to calculate the inter-cluster distances. We obtain most compact and separable granules (clusters) with small values of intra-cluster distances and large values of inter-cluster distances.

Validity indices **[5]** determine the quality of clustering by evaluating the compactness and separation of discovered clusters. Assessment of the most optimal result needs calculation of validity indices for different values of algorithm's parameter, what usually is the number of clusters.

In this project, we have implemented four of the most common validity indices – Dunn index, Davis-Bouldin (DB) index, Silhouette index and CDbw index. These indices have been discussed in the following sub-sections.

4.1: DUNN INDEX

The Dunn index **[6]**, introduced by J. C. Dunn in 1974,a metric for evaluating clustering algorithms, is an internal evaluation scheme, where the result is based on clustered data itself. Like all other validity indices, the aim of Dunn index is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well-separated, where the means of clusters are sufficiently far apart, as compared to the inter-cluster variance.

Higher the value of Dunn index, the better is the clustering. This index also has some drawbacks. As the number of clusters and dimensionality of the data increase, the computational cost also increases.

The Dunn index for K number of clusters is defined as :

$$\text{Dunn index (U)} = \min_{1<=i<=K} \left\{ \min_{1<=j<=K,\ i!=j} \left\{ \partial\,(X_i\,,\,X_j) \,/\, \max_{1<=k<=K}\ \{\Delta\,(X_k)\} \right\}\right\}$$

where,

$\partial\,(X_i\,,\,X_j)$ is the inter-cluster distance, that is, the distance between clusters X_i and $X_{j.}$

$\Delta\,(X_k)$ is the intra-cluster distance, that is, the distance within the cluster X_k .

ALGORITHM FOR DUNN INDEX EVALUATION

```
1:      Determine the number of clusters K from the already clustered file.
```

```
2:  Define intra→0, inter→MAX_VALUE (some big positive number).
3:  for i→1 to K
4:      Determine the average distance between all pairs of elements of this cluster.
        Let this distance be temp1.
5:      Modify intra = max(intra,temp1).
6:  repeat for all i
7:  for i=1 to K
8:      for p= 1 to no. of elements of this cluster
9:          for j=1 to K
10:             if i→j, continue.
11:             else find the average distance from p-th element to all the
                elements in cluster j. Let this distance be temp2. Modify
                inter=min(inter,temp2);
12:         repeat for all j
13:     repeat for all p
14: repeat for all i
15: Dunn index = inter/intra
```

Figure 5: Algorithm for Dunn index evaluation

4.2: DAVIES-BOULDIN (DB) INDEX

The Davies-Bouldin index, popularly known as DB index, introduced by David L. Davies and Donald W. Bouldin in 1979, a metric for evaluating clustering algorithms, is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset.

The DB index measure the average similarity between each cluster and its most similar one, and thus it is desirable to minimize this value. Thus, lower the value of DB index, better is the clustering. This index also has a drawback. A good value reported by this method does not imply the best information retrieval.

The DB index for K number of clusters is defined as:

$$\boxed{\text{DB index } (U) = 1/K \ \sum_{1<=i<=K} \max_{i!=j} \left\{ (\Delta (X_i) + \Delta (X_j)) / \partial (X_i, X_j) \right\}}$$

where,

$\partial(X_i , X_j)$ is the inter-cluster distance, that is, the distance between clusters X_i and $X_{j.}$

$\Delta (X_k)$ is the intra-cluster distance, that is, the distance within the cluster X_k .

```
ALGORITHM FOR DB INDEX EVALUATION
1:  Determine the number of clusters K from the already clustered file.
2:  Define Sum → 0.
3: for i → 1 to K
4:  Compute twice the average distance between all the points belonging
    to cluster $C_i$ and the centroid of cluster $C_i$ (centroid diameter linkage
    distance). This serves as $\Delta (X_i)$.
5:  for j → 1 to K, j != i
6:      Compute $\Delta (X_j)$ in the same way as $\Delta (X_i)$ (as in step 4).
7:      Calculate the distance between the centroids of clusters $C_i$ and
        $C_j$ (centroid linkage distance). This serves as $\partial (X_i , X_j)$.
8:      Calculate the ratio $R = (\Delta (X_i) + \Delta (X_j)) / \partial (X_i, X_j)$.
9:  repeat for all j
10: Find max(R) for a particular i;  Sum → Sum + max(R) for all i.
11: repeat for all i
12: DB index = Sum / K.
```

Figure 6: Algorithm for Db index evaluation

4.3: SILHOUETTE INDEX

Silhouette index (SI) was proposed by Kaufman and Rousseeuw to measure the strengths of clusters. It is used to interpret and validate the consistency within clusters of data. It is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

A high value of Silhouette index indicates that an object is well-matched to its own cluster and poorly matched to neighboring clusters.

For a given cluster C, this method assigns to each object "y_i" of C, a quantitative measure s_i. Let, a_i be the average distance of the object y_i to other objects in the cluster C and b_i is the average distance of y_i to objects in the nearest neighboring cluster. We define:

$$\boxed{s_i = (b_i - a_i) / \max(a_i, b_i)}$$

a_i actually stands for the average dissimilarity of i-th object to all other objects in the same cluster and b_i denotes the average dissimilarity of the i-th object with all objects in the closest cluster.

Now s_i lies between -1 and +1. If s_i is zero, then the object y_i has equal distances to its cluster and the nearest neighboring cluster. If the index is positive, then the object is closer to its cluster than other clusters and if it is negative, then the object has been wrongly assigned to its current cluster. The Silhouette index is defined as:

$$\boxed{SI = 1/\mathrm{card}(U) \sum_{y_i \in U} s_i}$$

where, U is the set of all points of the dataset and card(U) is the number of points in the entire dataset. SI is a measure of how tightly grouped all the points in the cluster are. It suffers from the limitations like assumption of distribution of clusters (mostly elliptical and spherical) or requirement of clusters of equal size and densities.

ALGORITHM FOR SILHOUETTE INDEX EVALUATION

```
1:  Determine the number of clusters K from the already clustered file.
2:  Initiate intra→0 , inter→MAX_VALUE(some big positive number),
    sum→0.

3: for i → 1 to K

4:  Determine the average distance between all pairs of elements of this cluster.
    Let this distance be temp1.

5:  intra → temp1

6:  for p= 1 to no. of elements of this cluster

7:      for j=1 to K

8:          if i → j, continue;
9:          else find the average distance from p-th element to all the
            elements in cluster j. Let this distance be temp2. modify
            inter=min(inter,temp2).
10:     repeat for all j

11:     sum → sum + (inter-intra) / max(inter, intra);

12:  repeat for all p

13: repeat for all i

14: Silhouette index = sum/N, N = total number of elements/objects in the dataset.
```

Figure 7: Algorithm for Silhouette index evaluation

4.4: CDbw INDEX

CDbw index is based on cluster compactness and separation. It is a product of two components – Separability (Sep) and Intra-density (intra-dens).

$$\boxed{CDbw(K) = Sep(K) * intra\text{-}dens(K)}$$

where, K is the number of clusters.

Separability is a measure of separation between clusters and is expressed by the following equation:

$$\boxed{Sep(K) = \{ \sum_{1<=i<=K} \sum_{1<=j<=K,\, j!=i} d(clos\_rep_i , clos\_rep_j) \} / (1 + inter\_dens(K))}$$

Separability is proportional to the sum of distances between the closest representative points *close_rep* from pair-wise clusters and inversely proportional to measure of density between clusters, *inter_dens*. Representative points are objects from training set selected by FARTHEST FIRST (FF) algorithm, the steps of which has been listed while discussing the algorithm for this index.

The density between clusters is expressed by the following equation. It is desirable to generate partitioning of the lowest inter-cluster density.

$$\boxed{Inter\_dens(K) = \sum_{1<=i<=K} \sum_{1<=j<=K\,,i!=j} (d(clos\_rep_i , clos\_rep_j)*density(u_{ij}) / \sum_{i,j}stdev)}$$

$\sum_{i,j}$stdev is the summation of standard deviation *stdev* of clusters C_i and C_j. The following equation is used to determine this value:

$$\boxed{stdev(C_i) = 1/ |C_i| \ \sqrt{(\sum (d\ (x, x`))^2\ )}}$$

where, x`is the centroid of cluster $C_i$, d(x, x`) is the distance between object x belonging to cluster C_i and x` and $|C_i|$ is the number of elements in this cluster.

density(u_{ij}) is the density of input objects around the point u_{ij}, which is the middle point on the line segment joining clos_rep$_i$ and clos_rep$_j$. It represents the percentage of points in the cluster C_i and the cluster C_j that belong to the neighborhood of u_{ij}. This neighborhood is defined to be a hyper-sphere with centre u_{ij} and radius the average standard deviation of the clusters between which the density is estimated. The expression of density(u_{ij}) is as follows:

$$\boxed{\text{density}(u_{ij}) = (\sum f(x, u_{ij})) / (|C_i| + |C_j|) \; ; \; x \in C_i \cup C_j}$$

The function f(x, u_{ij}) is defined as follows:

$$\boxed{f(x, u_{ij}) = 0 \text{ if } d(x, u_{ij}) > (\text{stdev}(C_i) + \text{stdev}(C_j))/2 \; ; \quad 1 \text{ otherwise}}$$

This completes the evaluation of separability part. Now we calculate the intra-density *intra-dens* which determines the average density within clusters and is defined as the percentage of points that belong to the neighborhood of of representative points of the considered clusters. The goal is to obtain significantly high intra-cluster density. intra-dens(K) is given by the following equation:

$$\boxed{\text{intra-dens}(K) = 1/K \sum_{1<=i<=K} 1/r \sum_{v_{ij} \in C_i} \text{density}(v_{ij}) / \text{stdev}(C_i)}$$

where, r is the required number of representative points in each cluster and density(v_{ij}) is given by:

$$\boxed{\text{density}(v_{ij}) = \sum_{x \in C_i} g(x, v_{ij})}$$

The function g(x, v_{ij}) is described by the following equation:

$$\boxed{g(x, v_{ij}) = 0 \text{ if } d(x, v_{ij}) > \text{stdev}(C_i) \; ; \quad 1 \text{ otherwise}}$$

This completes the evaluation process of Cdbw index. A maximum value of this index indicates the best clustering scheme. Like other validity indices, this index may also give inconclusive results as it provides only some guidelines.

```
ALGORITHM FOR CDbw INDEX EVALUATION

1:    Determine the number of clusters K from the already clustered file.
2:    Define sum1→0 and inter→0.
3:    For each cluster $C_i$, 1<=i<=K; find the required number, r, of Representative
      Points. Use the FARTHEST FIRST algorithm for this purpose:
          i>    Initially the cluster centroid is determined.
          ii>   Then the object located farthest from the centroid belonging to
                the same cluster is chosen as the first Representative point.
          iii>  Now the farthest points, within the same cluster, from the
                previously determined representatives are selected till we get r
                such representatives.
4: for i→ 1 to K-1, for j→ i+1to K; that is, for every distinct pair of clusters
5:    Compute the distance between the closest Representative points of the two
      clusters, $clos\_rep_i$ and $clos\_rep_j$ and let this distance be d.
6:    Compute the standard deviation of the two clusters, $stdev(C_i)$ and $stdev(C_j)$.
7:    Compute the cardinality of the two clusters, $|C_i|$ and $|C_j|$, respectively.
8:    Now find out the mid-point of $clos\_rep_i$ and $close\_rep_j$, $u_{ij}$ by co-ordinate
      geometry.
9:    Define S→0.
10:   for all points x belonging to $C_i$ U $C_j$
11:       let distance between x and $u_{ij}$ be d`
12:       if d` > $(stdev(C_i) + stdev(C_j)) / 2$, continue.
13:       else S→ S + 1.
14:   repeat for all x
15:   Define density($u_{ij}$) = S / ($|C_i| + |C_j|$).
```

```
ALGORITHM FOR CDbw INDEX EVALUATION (contd.)
16:     inter → inter + {(d * density($u_{ij}$)) / (stdev($C_i$) + stdev($C_j$)}
17:     sum1 → sum1 + d
18: repeat for all i and j
19:     define sep = sum1 / (1 + inter). sep represents separability.
20:     define sum2→0.
20: for i → 1 to K
21:     define sum3→0.
22:     Compute the standard deviation of cluster $C_i$ , stdev($C_i$) (or just use it if
        stored earlier).
23:     for all points v belonging to $C_i$
24:             for all other points x belonging to $C_i$
25:                     Compute distance, d`` between x and v.
26:                     if d`` > stdev($C_i$), continue.
27:                     else sum3 → sum3 +1.
28:             repeat for all x
29:     repeat for all v
30:     sum2 → sum2 + sum3 / (stdev ($C_i$) * r).
31: repeat for all i
32:     define intra → sum2 / K. intra stands for intra-density.
33:     Dunn index = sep * intra.
```

Figure 8: Algorithm for CDbw index evaluation

5. EMPIRICAL EVALUATION

EXPERIMENTAL SETUP

This experimental setup for this experiment has been described as follows:

<1> The experiments were carried out in a *HP* laptop with I3 processor.

<2> Datasets from 15 different domains have been gathered for this experiment (details have been discussed in the following sub-section).

<3> K-means, K-medians, K-medoids and Bisecting K-means algorithm have been employed to carry out the experiments, that is, to cluster the original dataset files.

<4> Given the iterative nature of these algorithms and the random initialization of centroids(or medoids), there is a possibility that these algorithms may get stuck in a local optimum. So it is recommended to run the algorithms using different initializations (50 times) of centroids or medoids and pick the results of the run that yielded the best (least) objective function (SSE/SE).

<5> The criteria we have applied to judge the performances of these 4 algorithms are the objective function (SSE/SE) values and the 4 validity indices – Dunn index, DB index, Silhouette index and the Cdbw index.

DATASETS

We have done the experiment on the original datasets for 15 domains, the files of which were collected from the data repository of University of California (UCI) at Ervine which are used for machine learning. These are listed as follows:

(i) echocardiogram (ii) ecoli (iii) heart Hungary (iv) glass (v) iris (vi) liver disorder (vii) machine (viii) new thyroid (ix)pima Indians (x) titanic (xi) thyroid (xii) vehicle (xiii) wine (xiv) wisconsin (xv) yeast

Each domain has two files: *.data and *.names. The .data files were used for clustering, the last column of each of these files being the class attribute. The .names files contain the descriptions of the data set. The first line gives the number and the values of the class attribute. Subsequent values give the values for discrete attributes or for the continuous types, it is marked as 'continuous'.

The .data files were first normalized (as per point <3> of EXPERIMENTAL SETUP) and then the different clustering algorithms were run on them and the results were stored in different files. Then the validity indices of all the clustered files were evaluated and further discussions have been done on the basis of these validity indices.

MISSING VALUE TREATMENT

Some of the .data files had attributes containing missing values (indicated by a ‘?’). These were taken care of in the following ways:

(i) For a discrete attribute having a missing value, the ‘?’ was replaced by a separate value, which is the smallest number greater than the maximum of the values present for that particular attribute. For example, if a discrete attribute is having values 1,2,3,4 and having a missing value in the .data file, then it has been replaced by 5 in that corresponding column.

(ii) If a continuous attribute is having missing values, then those are similarly replaced by a separate (and highest value, but not far higher values). For example, if a continuous attribute is having some missing values and the values present are in the range 0.10-0.45, then the missing values are replaced by values greater than 0.45 but they should not be much larger than 0.45.

SCALING THE ORIGINAL DATAFILES

Since the applied clustering algorithms use distance-based measurements to determine the similarity between data points, it’s recommended to normalize (or scale) the data after missing value treatment to have a mean of zero and a standard deviation of one since almost always the features in any dataset would have different units of measurements such as age versus income where the values differ in order of powers of 10 and thus it is required to bring them down within the same range. This normalization may be done by scaling the data using the following formula:

$$\text{scaled}(a_{ij}) = \{\max(a_{ij}) - a_{ij}\} / \{\max(a_{ij}) - \min(a_{ij})\},$$

where the maximum and minimum is calculated for a particular attribute (or,column). Here all the values will be in the range 0-1.

RESULTS AND DISCUSSIONS

Table 1 shows the cluster distribution of the 15 datasets formed by the clustering algorithms against the original cluster distribution. Further calculations and discussions have been done on these clustered files.

NAME of DATASET	NO. of CLUSTERS	ORIGINAL CLUSTER DISTRIBUTION	CLUSTERING BY 4 CLUSTERING ALGORITHMS			
			K-means	K-medians	K-medoids	Bisecting K-means
echocardiogram	3	24,58,50	31,76,25	16,85,31	44,65,23	44,1,87
ecoli	8	143,77,2,2,35, 20,5,52	40,61,48,21, 10,100,0,56	37,10,36,42, 60,58,34,59	80,20,10,32, 95,38,31,30	104,50,117,0, 8,47,0,10
glass	7	70,76,17,0, 13,9,29	41,3,6,25, 17,24,98	27,3,26,20, 23,19,96	7,15,15,14, 21,112,30	15,35,0,43, 0,3,118
heart Hungary	5	37,26,28,15, 188	66,91,20,32, 85	64,74,75,54, 27	63,76,57,73, 25	113,32,103,0, 46
iris	3	50,50,50	61,50,39	61,50,39	71,28,51	50,61,39
liver disorder	2	145,200	293,52	215,130	143,202	294,51
machine	30	1,9,2,2,2,8,6,5, 9,6,7,5,1,3,7,7, 13,32,6,6,1,19, 13,3,3,5,12, 13,1,2	6,0,2,0,0,1,1, 2,0,2,0,0,22,0, 0,0,0,0,1,15, 16,29,21,1,18, 10,6,18,19,19	0,20,0,18,2, 0,0,0,0,0,10, 20,58,18,0, 0,0,0,0,0,8, 0,0,1,0,0,0, 2,1,51	8,14,4,2,23,3, 1,18,1,8,2,5, 10,6,5,2,3,4, 20,3,5,5,2,9, 8,1,8,1,23,5	1,31,3,0,0,0,0,0, 2,42,0,2,5,61,0, 40,2,0,0,0,0,0, 19,0,0,0,0,0,1,0
new thyroid	3	149,35,30	173,23,18	25,165,24	64,73,77	18,23,173
pima Indians	2	268,500	515,253	293,475	166,602	517,251
titanic	2	711,1490	1731,470	1731,470	1731,470	454,1747
thyroid	2	65,150	18,197	26,189	189,26	39,176
vehicle	4	28,20,26,20	25,46,2,21	24,37,14,19	9,18,30,37	2,30,27,35
wine	3	59,71,48	55,63,60	75,62,41	88,87,3	70,48,60
wisconsin	2	241,458	233,466	220,479	486,213	233,466

yeast	10	225,420,383,43,148,45,26,13,18,3	12,0,1,14,494,448,216,0,0,149	0,318,256,131,140,233,14,0,107,135	155,164,142,157,135,240,59,60,205,17	0,151,568,5,3,98,171,0,251,87

TABLE 1: Table showing the number of clusters, the original cluster distribution and the cluster distributions of the 15 datasets obtained by the 4 algorithms

Tables 2, 3, 4, 5 and 6 displays the values of SSE/SE, Dunn index, DB index, Silhouette index and CDbw index, respectively, for each of the clustered files of each dataset domain. These have been recorded as follows.

SRL NO	NAME of DATASET	SSE value for K-means	SE value for K-medians	SE value for K-medoids	SSE value for Bisecting K -means
1	echocardiogram	57.4649	165.7268	190.5249	265.4159
2	ecoli	18.9476	127.8142	158.5382	69.9515
3	glass	16.3895	99.5555	123.4905	125.3349
4	heart Hungary	233.8697	274.5419	566.1273	434.9802
5	iris	6.9981	47.7763	59.3376	127.0310
6	liver disorder	29.9644	170.6223	195.2965	487.8802
7	machine	3.0426	45.8176	32.7147	19.7253
8	new thyroid	10.6019	64.3036	78.6589	129.3018
9	pima Indians	121.2575	622.0416	723.8670	1158.4951
10	titanic	337.1866	704.9974	706.9948	1515.7604
11	thyroid	16.3594	77.0727	79.9701	143.7319
12	vehicle	29.4820	153.5674	189.9796	145.1252
13	wine	57.0407	211.1984	269.5146	139.2590

14	wisconsin	243.1924	729.2125	777.3340	1618.3469
15	yeast	73.4103	509.9064	558.4467	133.5165

TABLE 2: Table showing the values of SSE/SE calculated for the clustered files, clustered by different clustering algorithms, of the 15 datasets.

SRL NO	NAME of DATASET	Dunn index value for K-means	Dunn index value for K-medians	Dunn index value for K-medoids	Dunn index value for Bisecting K -means
1	echocardiogram	0.1312	0.1222	0.065	0.0883
2	ecoli	0.1612	0.2213	0.0452	0.1576
3	glass	0.0427	0.0131	0.0343	0.0219
4	heart Hungary	0.0255	0.0232	0.0237	0.0243
5	iris	0.5186	0.4363	0.0723	0.5186
6	liver disorder	0.0273	0.0153	0.0476	0.0282
7	machine	0.0432	0.0001	0.0173	0.0634
8	new thyroid	0.1038	0.0632	0.0803	0.1033
9	pima Indians	0.0235	0.0211	0.0213	0.0235
10	titanic	0.2912	0.2918	0.0448	0.1310
11	thyroid	0.1714	0.1082	0.0811	0.0823
12	vehicle	0.0431	0.0328	0.0311	0.0420
13	wine	0.0400	0.0281	0.0143	0.0498
14	wisconsin	0.3201	0.3282	0.3354	0.3214

15	yeast	0.0660	0.0701	0.0502	0.0033

TABLE 3: Table showing the values of Dunn index calculated for the clustered files, clustered by different clustering algorithms, of the 15 datasets.

SRL NO	NAME of DATASET	DB index value for K-means	DB index value for K-medians	DB index value for K-medoids	DB index value for Bisecting K -means
1	echocardiogr am	2.4631	1.1970	88.5616	443.5698
2	ecoli	5548.2346	4298.2876	18.4948	2233.1768
3	glass	0.3575	12.0105	0.4271	4978.9650
4	heart Hungary	6.9340	12.3349	15.4430	465.3394
5	iris	0.2991	0.1211	0.7553	0.1348
6	liver disorder	2.7155	0.5691	0.3724	2.7946
7	machine	4378.2981	19.3095	3.4992	27.9980
8	new thyroid	8.4009	0.5805	0.5055	0.3308
9	pima Indians	2.7786	1.0457	0.7659	2.8148
10	titanic	2.3734	0.3996	0.0073	0.6547
11	thyroid	0.2624	0.3916	0.4393	0.5716
12	vehicle	20.0730	0.7887	19.6935	0.3294
13	wine	0.5200	9.3603	5800.7299	0.4463

14	wisconsin	0.4240	0.4974	0.4901	0.3441
15	yeast	3879.1193	1596.7133	26.8377	0.0102

TABLE 4: Table showing the values of DB index calculated for the clustered files, clustered by different clustering algorithms, of the 15 datasets.

SRL NO	NAME of DATASET	Silhouette index value for K-means	Silhouette index value for K-medians	Silhouette index value for K-medoids	Silhouette index value for Bisecting K -means
1	echocardiogr am	-0.2281	-0.5302	0.0800	-0.3475
2	ecoli	-1.0000	-1.0000	-0.5488	-1.0000
3	glass	-0.1096	-1.0000	-1.0000	-1.0000
4	heart Hungary	-0.1888	-0.5598	-0.6347	-0.4256
5	iris	0.6955	0.6913	-0.2205	0.6955
6	liver disorder	0.3281	0.3180	-0.3485	0.3194
7	machine	-0.2433	-0.2380	-1.0000	-0.7789
8	new thyroid	-0.1078	0.1022	-0.3176	-0.1070
9	pima Indians	-0.1259	-0.0659	-0.0584	-0.1296
10	titanic	0.2742	0.2742	0.1565	-0.2289
11	thyroid	-0.3273	-0.1427	-0.0960	0.6740
12	vehicle	-0.6213	0.3613	0.3452	-0.6737
13	wine	0.2892	0.1791	-0.5733	0.2443

14	wisconsin	0.7960	0.7696	0.7480	0.7930
15	yeast	-1.0000	-1.0000	-0.6480	1.0000

TABLE 5: Table showing the values of Silhouette index calculated for the clustered files, clustered by different clustering algorithms, of the 15 datasets.

SRL NO	NAME of DATASET	CDbw index value for K-means	CDbw index value for K-medians	CDbw index value for K-medoids	CDbw index value for Bisecting K -means
1	echocardiogr am	118.3257	139.0842	123.5966	289.0999
2	ecoli	1884.9055	1254.3560	1722.642 2	958.6659
3	glass	1441.7018	1370.5684	1244.485 8	945.8370
4	heart Hungary	378.6712	629.7129	777.2291	453.2209
5	iris	772.5754	776.0444	587.3133	772.5754
6	liver disorder	569.9676	934.1092	1414.469 0	570.0220
7	machine	6362.6948	9052.8896	1271.371 6	3563.2192
8	new thyroid	1294.8491	797.9279	403.3952	1310.3871
9	pima Indians	625.4151	187.9039	914.0716	626.3276
10	titanic	12585717.00 00	12585717.00 00	467.0019	5088610.00 00
11	thyroid	682.5680	568.5828	776.2210	1170.9244
12	vehicle	97.8766	39.4372	85.7415	76.9382

13	wine	301.7116	105.5362	288.8589	247.5098
14	wisconsin	13621.3682	13319.6855	14436.2412	13344.6182
15	yeast	1931.5292	6261.2900	12058.1662	355.4908

TABLE 6: Table showing the values of CDbw index calculated for the clustered files, clustered by different clustering algorithms, of the 15 datasets.

Tables 2, 3, 4, 5 and 6 show the validity indices calculated on the clustered files of the 15 datasets. Although the contents of these tables are interesting in themselves, they have been summarized in the following tables: Tables 7, 8, 9, 10 and 11, respectively. These tables compare the 4 clustering algorithms on the basis of these index values. The values have been compared for each index against 3 criteria. These have been mentioned and discussed in brief as follows:

(i) ARITHMETIC MEAN: The arithmetic mean for a particular index for a particular clustering algorithm has been calculated by taking the average of the values of that index for each dataset file clustered by that particular clustering algorithm.

(ii) ∑ OF SCORE: There are 4 algorithms. The algorithm which yields the best optimum value of a particular index gets a score of 4, the next optimum gets 3, the next one 2 and the worst gets 1 for that index. If there is a tie at a position, then the score for that position is distributed among the tied algorithms in such a way that the total score of all the algorithms for that dataset for that index remain 10. For example, if for a particular dataset and a particular index, K-means yields the best value, followed by K-medians and K-medoids yielding the same value, and Bisecting K-means yielding the worst value. Then K-means gets a score of 4 as it gives the best value and Bisecting K-means gets a score of 1 as it gives the worst value. Now to have a total score of 10, the total of the scores of K-medians and K-medoids should be 5. So, each of them gets a score of 5/2=2.5. ∑ of score for an algorithm for a particular index is given by the summation of the scores obtained by that algorithm for all the datasets for that index.

(iii) ∑ OF TOP SCORE: In this criterion, for a particular index, the clustering algorithm which gives the optimal result for a particular dataset gets the top score of 1 for that dataset and the remaining algorithms get 0 for that dataset for that index. If there is a tie for the best clustering, then the algorithms causing the tie has the score divided among themselves equally. For example, if for a particular index for a particular dataset, two clustering methods give the best index value, then both get ½=0.5 for that index and dataset. For a particular index, ∑ of top score for an algorithm is given by the summation of the top scores obtained by that algorithm for all the datasets for that index.

Criteria (for SSE/SE) ↓ Algorithms →	K-means	K-medians	K-medoids	Bisecting K-means
Arithmetic Mean	83.6805	266.9436	314.0530	434.2571
∑ of score	60	39	25	26
∑ of top score	15	0	0	0

TABLE 7: Summary of TABLE 2

The points revealed by TABLE 7 regarding the performances of the four clustering algorithms are listed as follows:

(1) Minimum value of SSE/SE indicates the best clustering. So, according to the Arithmetic Mean values of SSE/SE; K-means is the best clustering method, that is, produces the best (least) values of SSE/SE. Then comes K-medians, followed by K-medoids and lastly, Bisecting K-means algorithm.

(2) The ∑ of score parameter reveals that K-means has earned the highest score. Then comes K-medians, followed by Bisecting K-means and lastly, K-medoids. Now, higher the score, the better is the clustering technique.

Hence, K-means is the best and K-medoids the worst according to this parameter.

(3) The ∑ of top score reveals that K-means has given the best value for 15 times, that is basically K-means has given the best values of SSE/SE for all the datasets. Rest all the clustering algorithms have not opened their accounts for this parameter.

Criteria (for Dunn index) ↓ Algorithms →	K-means	K-medians	K-medoids	Bisecting K-means
Arithmetic Mean	0.1339	0.1183	0.0643	0.1106
∑ of score	47	35	30	38
∑ of top score	6	3	3	3

TABLE 8: Summary of TABLE 3

The points revealed by TABLE 8 regarding the performances of the four clustering algorithms are listed as follows:

(1) The maximum value of Dunn index signifies the best clustering. So, the Arithmetic Mean of the Dunn index values indicates that K-means is the best clustering technique with respect to Dunn index. Then comes K-medians in order, followed by Bisecting K-means and ultimately, K-medoids.

(2) The ∑ of score indicates that K-means has received the highest score, followed by Bisecting K-means, K-medians, and lastly K-medoids. This is also the order of their performance based on this criterion of Dunn index.

(3) The ∑ of top score reveals that K-means has obtained the highest top score of 6, followed by the other algorithms each with a top score of 3. Now higher the value of this parameter, the better the clustering as more is the number of times the algorithm gives the best value. Hence, K-means turns out to be the best algorithm with respect to this parameter of Dunn index.

Criteria (for DB index) ↓ Algorithms →	K-means	K-medians	K-medoids	Bisecting K-means
Arithmetic Mean	923.5502	396.9071	398.4682	543.8320
∑ of score	33	40	41	36
∑ of top score	3	2	5	5

TABLE 9: Summary of TABLE 4

The points revealed by TABLE 9 regarding the performances of the four clustering algorithms are listed as follows:

(1) Minimum value of DB index indicates the best clustering. Hence, from the Arithmetic Mean values of this index, we infer that K-medians is the best clustering technique, followed by K-medoids, then Bisecting K-means and lastly K-means, giving the worst clustering with respect to DB index.

(2) The ∑ of score parameter reveals that K-medoids has earned the highest score, followed by K-medians, then Bisecting K-means and lastly K-means. According to this parameter, K-medoids is the best, K-medians comes in the second place, then comes Bisecting K-means and K-means.

(3) The ∑ of top score reveals that both K-medoids and Bisecting K-means have got the highest value for this criterion, that is, 5. K-means has got 3 and K-medians received 2. Now higher the value of this parameter, the better is the performance of that algorithm. Thus, we observe a different order of performance of algorithms.

Criteria (for Silhouette index) ↓ Algorithms →	K-means	K-medians	K-medoids	Bisecting K-means
Arithmetic Mean	-0.1046	-0.1227	-0.2744	-0.0643
∑ of score	43.5	37	32	37.5
∑ of top score	6	3.5	3	2.5

TABLE 10: Summary of TABLE 5

The points revealed by TABLE 10 regarding the performances of the four clustering algorithms are listed as follows:

(1) Maximum value of Silhouette index indicates the best clustering. Thus, the Arithmetic Mean values reveal that Bisecting K-means is the most efficient algorithm, followed by K-means, then K-medians and lastly, K-medoids.

(2) From the ∑ of score values, we observe that K-means is the top scorer, followed by Bisecting K-means, then K-medians and K-medoids. Thus, this

criterion claims K-means to be the best algorithm; followed by Bisecting K-means, K-medians and K-medoids in order of their performances.

(3) From the ∑ of top score values, we observe that K-means has received the highest value of this parameter, then comes K-medians, then K-medoids and then Bisecting K-means. As higher the value of this parameter, better the performance; so the order mentioned above is the decreasing order of performances of these algorithms.

Criteria (for CDbw index) ↓ Algorithms →	K-means	K-medians	K-medoids	Bisecting K-means
Arithmetic Mean	841053.4107	841410.2753	2438.0537	340886.3224
∑ of score	40	35.5	40	34.5
∑ of top score	4.5	2.5	5	3

TABLE 11: Summary of TABLE 6

The points revealed by TABLE 11 regarding the performances of the four clustering algorithms are listed as follows:

(1) To determine the most effective clustering, we need the maximum value of CDbw index. Hence, from the Arithmetic Mean values of this index, we infer that K-medians is the best clustering method, followed by K-means, Bisecting K-means and lastly K-medoids.

(2) The ∑ of score values reveal that both K-means and K-medoids are the top scorers, followed by K-medians and then Bisecting K-means. Thus, according to this criterion of CDbw index, K-means and K-medoids are the joint “winners”, followed by K-medians and Bisecting K-means in the last place.

(3) The ∑ of top score parameter reveals that K-medoids has attained the highest top score of 5 and is, thus, the best according to this criterion, followed by K-means, then Bisecting K-means and ultimately K-medians, in decreasing order of their performances.

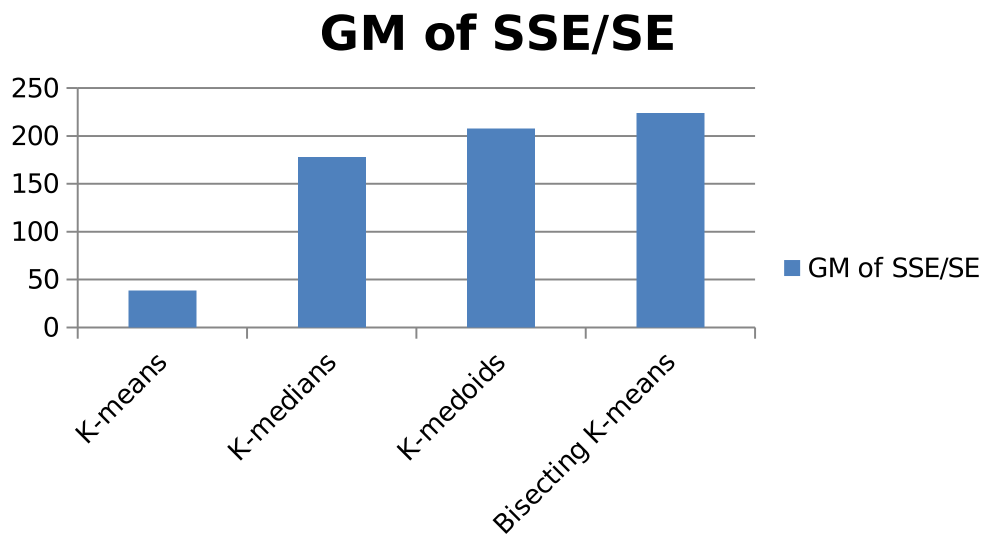
From the above tables (TABLE 7, 8, 9, 10 and 11) and the points revealed by the entries to these tables, we have made a few conclusions which are listed as follows:

(i) All the three criterion, that is, Arithmetic Mean, ∑ of score and ∑ of top score do not indicate the same order of performance of clustering algorithms.

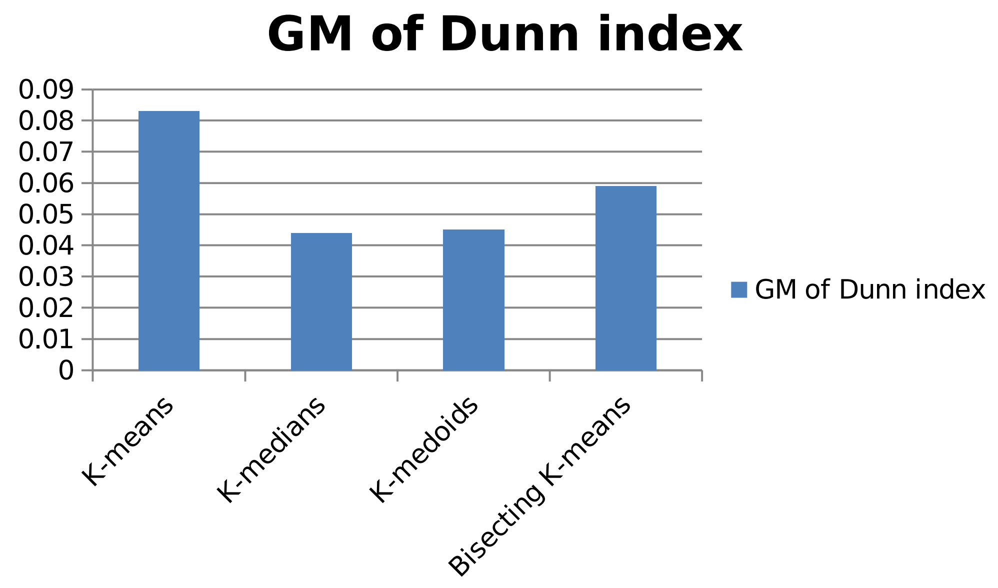
(ii) Arithmetic Mean values of an index depend directly on the magnitude of the values of that index given by a clustering algorithm. The ∑ of score values depends directly on the order of the values given by different algorithms for a particular dataset and does not do the calculation directly on the magnitude of values. However, in ∑ of score, all the values have some score according to the order of their magnitudes. Now ∑ of top score parameter gives credit only to the best values or the optimum values and there is absolutely no credit for the other values.

(iii) Thus, it is clear that these criteria may not reveal the same order of performance of the clustering algorithms. The difference of the orders is attributed to the different ways in which these criteria work. The algorithm with optimum value of Arithmetic Mean may not have the best ∑ of score. It may so happen (as has been the case with TABLE 9, 10, 11) that the algorithm with the worst Arithmetic Mean may have the best ∑ of top score and vice-versa. This is due to fact that the values may be widely separated in some cases and may be very closely spaced in some other cases. So, the variation in magnitude is not uniform but the scores do vary uniformly as scores depend only on the order of values.

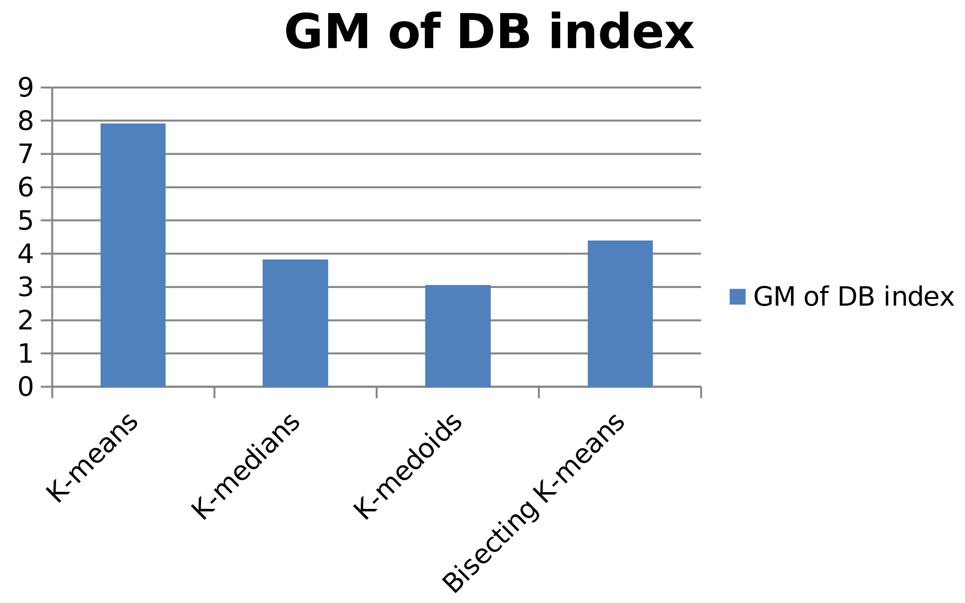
Now, the Arithmetic Mean values shown in tables 7, 8, 9, 10 and 11 are meaningful if the datasets are from the same domain. However, this is not the thing in our case. In such cases when the datasets are from different domains, Geometric Mean becomes more meaningful. Thus, we have plotted the Geometric Mean values for each clustering algorithm for each index in the following graphs (graphs 1, 2, 3, 4 and 5 show the geometric mean values of each clustering algorithm for SSE/SE, Dunn index, DB index, Silhouette index and CDbw index, respectively).



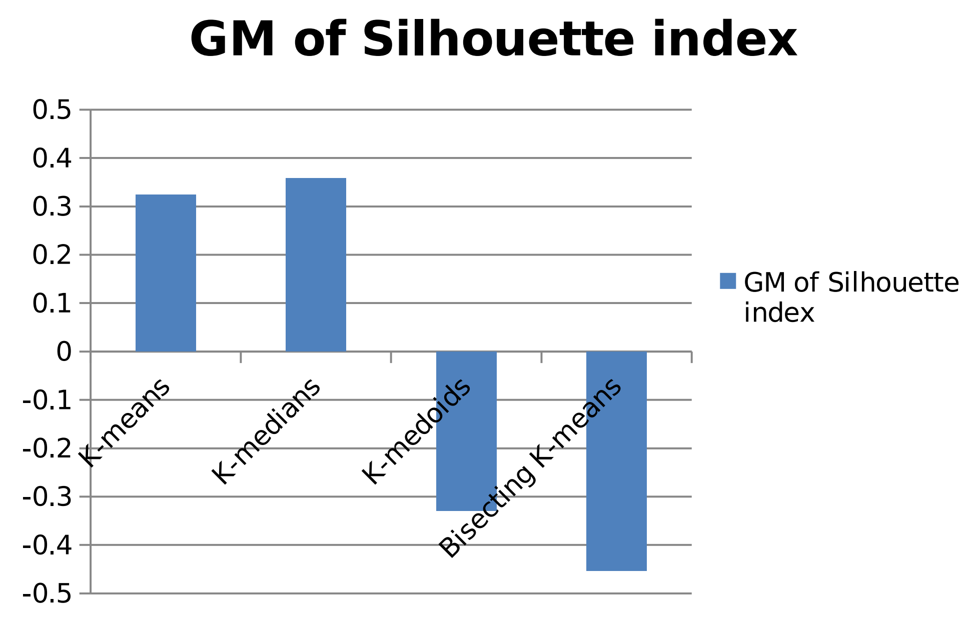
GRAPH 1: Geometrical Mean of SSE/SE values



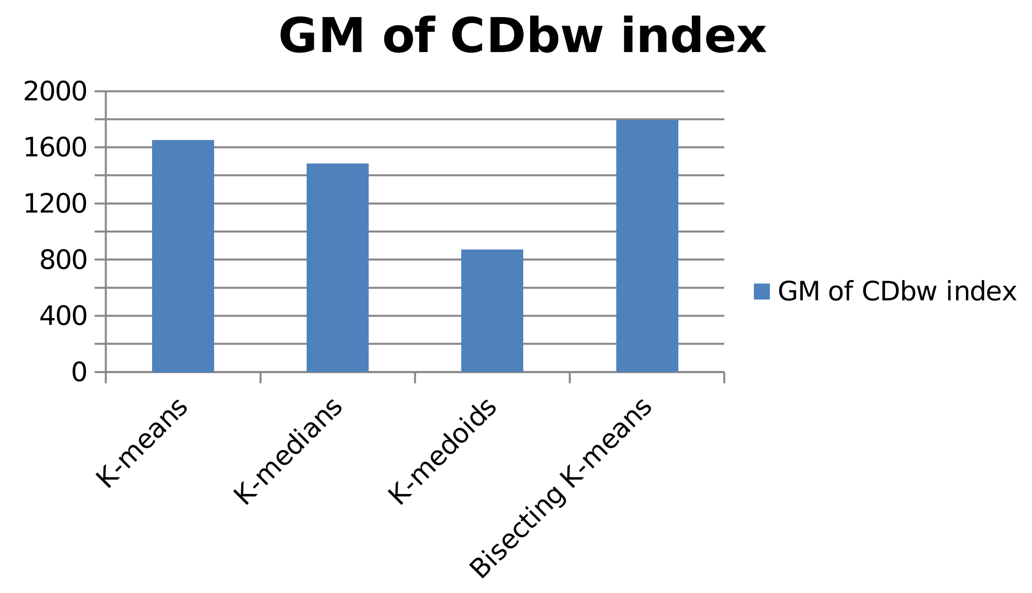
GRAPH 2: Geometrical Mean of Dunn index values



GRAPH 3: Geometrical Mean of DB index values



GRAPH 4: Geometrical Mean of Silhouette index values



GRAPH 5: Geometrical Mean of CDbw index values

Inference from the Graphs of Geometric Means:

1. GRAPH 1 contains the GM of SSE/SE values for each algorithm. The graph reveals that K-means has the best performance with respect to SSE/SE, then comes K-medians, then K-medoids and lastly Bisecting K-means.
2. GRAPH 2 displays the GM of Dunn index values for each algorithm. The graph reveals that K-means has the best performance, followed by Bisecting K-means, K-medoids and K-medians in order of their performances.
3. GRAPH 3 contains the GM of DB index values and indicates that K-medoids has the best performance, followed by K-medians, then Bisecting K-means and lastly K-means with respect to DB index.
4. GRAPH 4 reveals the GM of Silhouette index values and from this graph, we may conclude that K-medians is the best algorithm, followed by K-means, then K-medoids and lastly Bisecting K-means as per this index.
5. GRAPH 5 shows the GM of CDbw index and reveals that Bisecting K-means produces the best results, followed by K-means, then K-medians and lastly K-medoids in order of their performances with respect to this index.

6. CONCLUSION

Our programs create data file for the datasets belonging to each of the 15 domains, chosen for our experiment, with appropriate cluster labels for each of the four clustering algorithms. The comparative performances of these algorithms have also been measured using four validity indices – Dunn index, DB index, Silhouette index and CDbw index; and an objective function (SSE/SE). The performance of these algorithms have been measured mainly on the basis of four criteria for each of these indices – Arithmetic Mean, ∑ of score, ∑ of top score (via tabular representation) and Geometric Mean (via graphical representation). This project assumes importance in the present context because in this century when the entire world has turned into a global village and internet has become an indispensable part of our lives, data has a very important role to play. The recent outbreak of covid-19 has once again proved that digitalization is going to be the future of the world, thus magnifying the importance of digital data. So clustering of data to extract meaning out of it is even more important. Measuring the quality of clustering is equally significant in this regard.

7. REFERENCES

1. MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press. pp. 281–297. *MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.*
2. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice-Hall, 1988.
3. Kaufman, L. and Rousseeuw, P.J., *Clustering by means of Medoids* , in Statistical Data Analysis Based on the L1- Norm and Related Methods(1988) , edited by Y. Dodge, North-Holland, 405–416.
4. [reference for bisecting K-means]
5. [reference for validity indices]
6. Dunn, J. C., *Well-separated clusters and Optimal Fuzzy Partitions ,* Journal of Cybernetics(1974), pp. 95-104. ISSN 0022-0280.