CHAPTER 4

USING ATTRIBUTE WEIGHTAGE FOR CLUSTERING AND LOW DENSITY CALCULATION FOR FURTHER BISECTING

4.1 INTRODUCTION

Data clustering is an important technique for data analysis which can be used to discover the similarity or dissimilarity between groups of items in a dataset such that items in one group are more similar than the other groups and vice versa. K_Means algorithm is the most popular partition based algorithm which is widely used in data clustering. A lot of algorithms have been proposed for data clustering using K_Means algorithm due to its simplicity, efficiency and ease convergence. K_Means is one of the oldest and most commonly used clustering algorithms. It is a prototype based clustering technique defining the prototype in terms of a centroid which is considered to be the mean of a group of points and is applicable to objects in a continuous n-dimensional space.

4.2 INITIAL CENTROID SELECTION

The K_Means algorithm involves randomly selecting K initial centroids where K is a user defined number of desired clusters. Each point is then assigned to the closest centroid and the collection of points close to a centroid form a cluster. The centroid gets updated according to the points in the cluster and this process continues until the points stop changing their clusters. It is known that the performance of the K-Means clustering algorithm depend highly on initial cluster centers. Generally initial cluster centers are selected randomly, so that the algorithm could not lead to the unique result.

Different runs of the algorithm can produce poor results when the initial centroids are chosen randomly. It is important to realize that the choice of the initial centroid has a huge effect on the final result. Solutions to this problem are,

- Multiple Runs: To perform multiple runs, each time a different set of initial centroids are randomly chosen initial centroids and then the set of clusters are selected with the minimum SSE. The problem with this method is that it doesn’t always work.
A sample of points are taken and clustered using a hierarchical clustering technique and then K clusters are extracted from this clustering and the centroids are then used as the initial centroids for the K-Means algorithm. The downside to this method is that it works only if the sample is relatively small and K is relatively small in comparison to the sample size.

The first point is to be selected at random for the first centroid and then for all the other initial centroids (k-1) select a point that is the farthest from any of the initial centroids that have already been selected. By doing this a set of initial centroids that is randomly selected is obtained and separated. The downside to this method is that it ends up selecting an outlier as a centroid.

Upgrading the algorithm and using the bisecting K-Means algorithm which is less susceptible to initialization issues.

In order to determine which centroid is the closest to a particular data point, a proximity measure is to be used. There are several proximity measures in use and typically one is chosen, based on the data type then it is tried to cluster. Manhattan, Euclidean, cosine and Bregman divergence are all proximity measures that help to determine which cluster a point should be assigned to. The Euclidean is commonly used.

The K-Means basic algorithm creates a couple of additional issues that must be considered and in some situations resolved in order to provide a realistic output. Empty Clusters occur when no points are assigned to a centroid and it also does not re-calculate the centroid value because no points are being used and so essentially will have an output with k-1 cluster. Outliers are points that are far away from the centroid and they can unduly influence centroid values and in turn this can skew cluster grouping and increase the amount of time needed to find an optimal solution. Outliers need to be discovered and eliminated beforehand so as to avoid these complications. It is important to note that not all outliers need to be removed in some situations like for data compression where every point must be clustered. There are a number of approaches used to avoid and remove outliers.
In this research work an initialization method is proposed for K-Means algorithm. The proposed method is used to generate the initial cluster centers rather than random or user specified cluster centers. Random selection is the common procedure but resulting clusters can produce poor quality clusters.

The centroid selection method given below is used to generate the initial cluster centers rather than random or user specified cluster centers.

\[
A = \frac{X(MAX) - X(MIN)}{K} \quad \text{------------------------------------------ (Equ. 4.1)}
\]

The generalization of the equation can be written as

\[
A_i = \frac{\text{MAX}(X_i) - \text{MIN}(X_i)}{K} \quad \text{------------------------------------------ (Equ. 4.2)}
\]

The Equation 4.2 is used to calculate the value of the variable M that specifies the range of initial cluster centers but not given the cluster centers where \( k \) is the number of clusters and \( \text{Max} \) is the maximum value of the attribute and \( \text{Min} \) is the minimum value of the attribute. The cluster centers for K-Means algorithm are generated using the given equation.

\[
C_k = \text{MIN}(X_i) + (K-1)A_i \quad \text{------------------------------------------ (Equ. 4.3)}
\]

This method is based on the algorithm developed by Hatamlou to obtain the most favorable cluster points but this method is used here to generate the initial cluster centers for K-Means algorithm. It is also noticed that the proposed method takes less time in execution compared to most of the algorithms.

4.3 INTRODUCTION OF ATTRIBUTE

An attribute describes the facts, details or characteristics of an entity. A single data item is related to a database object. The database schema associates one or more attributes with each database entity which is also known as field, column.
In general, an attribute is a property or characteristic. Color, for example, is an attribute of hair. In using or programming computers, an attribute is a changeable property or characteristic of some component of a program that can be set to different values.

4.3.1 Different types of attribute

A useful way to specify the type of an attribute is to identify the properties of numbers that correspond to the underlying properties of the attribute. For example, an attribute such as long has many of the properties of numbers. It makes sense to compare and order object by length, as well as to talk about the differences and the ratio of length. The following properties (operations) of numbers are typically used to describe attributes.

1) Distinctness = AND ?
2) Order <, =, > and =
3) Addition + and –
4) Multiplication * and /

Given these properties, four types of attributes can be defined: nominal, ordinal, interval and ratio. Each attribute type possesses all of the properties and operation of the attribute type about it. Consequently, any property or operation that is valid for nominal, ordinal, interval attribute is also valid for ratio attribute. In other words, the definition of the attribute type is cumulative and however this does not mean that the operation which is appropriate for one attribute type is appropriate for the attribute types.

Among the four types Nominal and Ordinal comes under Categorical type i.e Qualitative and the remaining two types interval and ration falls under the Numeric type i.e Quantitative.

Nominal : The values of nominal attribute are just different names i.e nominal values provide only enough information to distinguish one object from another(=, ?). Example for this type of attribute is zip codes, employee ID numbers, eye color, gender etc. Operations performed on this type of attribute is mode, entropy, contingency, correlation.

Ordinal : The values of an ordinal attribute provide enough information to order objects.(<,>).Examples are hardness of minerals(good, better, best),grades, street numbers. Operation performed on this type is median, percentiles, rank correlation run tests and sign tests etc.
Interval: For interval attributes the differences between values are meaningful i.e. a unit of measurement exists. (±, -). Some examples are calendar dates, temperature in Celsius or Fahrenheit. Mean, Standard deviation, Pearson’s correlation, $t$ and $F$ tests operation can be performed on this type of attribute.

Ratio: For ratio variables, both differences and ratios are meaningful. (×, /). Temperature in Kelvin, Monetary quantities, counts, age, mass, length, electrical current are some of the examples. Operations performed on this type are geometric mean, harmonic mean and percent variation [80].

4.3.2 Describing Attributes by the Number of Values

An independent way of distinguishing between attributes is by the number of values they can take.

Discrete: A discrete attribute has a finite or countably infinite set of values. Such attributes can be categorical, such as zip codes or ID numbers or numeric, such as counts. Discrete attributes are often represented using integer variables. Binary attributes are a special case of discrete attributes and assume only two values. E.g true or false, yes/no, male/female, or 0/1. Binary attributes are often represented as Boolean variables or as integer variables that only take the values of 0 or 1.

Continuous: A continuous attributes is one of whose values are real numbers. Examples include attributes such as temperature, height or weight. Continuous attributes are typically represented as floating-point variables. Practically, real values can only be measured and represented with limited precision.

In theory, any of the measurement scale types—nominal, ordinal, interval and ratio could be combined with any of the types based on the number of attribute values—binary, discrete and continuous. However, some combinations occur only infrequently or do not make much sense. For instance, it is difficult to think of a realistic data set that contains a continuous binary attribute. Typically, nominal and ordinal attributes are binary or discrete, while interval and ratio attributes are continuous. However, count attributes, which are discrete, are also ratio attributes.
Asymmetric Attributes

For asymmetric attributes, only the presence of a non-zero attribute value is regarded as important. Let a dataset be considered where each object is a student and check each attribute records, whether or not a student search for a particular course at a university. For a specific student, an attribute has a value of 1 if the student took the course associated with that attribute and a value of 0 otherwise. Because students take only a small fraction of all available courses, most of the values in such a data set would be 0. Therefore, it is more meaningful and more efficient to focus on the non-zero values. To illustrate, if students are compared on the basic of the courses they don’t take, then most student would be seen very similarly, atleast if the number of courses is large. Binary attributes where only non-zero values are important are called asymmetric binary attributes. This type of attribute is particularly important association analysis. It is also possible to have discrete or continuous asymmetric feature. For instance, if the number of credit associated with each courses is recorded then the resulting dataset will consist of asymmetric or continuous attributes [80].

4.3.3 Discretization of Attributes

Discrete values have important role in data mining and knowledge discovery. Many studies have shown that induction takes can benefit from discretization rules whereas discrete values are normally shorter and hence easier to understand. Discretization can lead to improved predictive accuracy. Furthermore, many induction algorithms found in the literature require discrete features. All these prompt the researches and practitioners to discretize continuous features before or during a machine learning or data mining task.

Data usually comes in a mixed format: normal, discrete, and/or continuous. Discrete and continuous data types have orders among the values, while nominal values do not possess any order amongst them. Discrete data are spaced out with intervals in a continuous spectrum of values. While the number of continuous values for an attribute can be infinitely large, the number of discrete values is often few or finite. These two types of values make a difference in learning classification tree/rule. When a decision tree is induced, handling continuous values make the induction process more complicated than using the discretized values.

There are many advantages of using discrete values over the continuous ones.

1) Data can also be reduced and simplified through discretization.
2) For both users and experts, discrete features are easier to understand, use, and explain.
3) Discretization makes learning more accurate and faster.
4) Results obtained using discrete features are usually more compact, shorter, and more accurate than using continuous ones. Hence the results can be more closely examined, compared, used, and reused.

Discretization is a process of quantizing continuous attributes. The success of discretization can significantly extend the borders of many learning algorithms.

Discretization is dividing values of numerical attributes into intervals and using the interval to replace the original value. Discretization of continuous attributes not only broadens the scope of the number of data mining algorithms able to analyze data in discrete form, but can also dramatically increase the speed at which these tasks can be performed.

4.4 DISCRETIZATION TECHNIQUES

4.4.1 Unsupervised Discretization

The simplest means of discretizing continuous features involves a class blind approach where only knowledge of the feature to be discretized is required. Here the user specifies the number of intervals or bins.

4.4.1.1 Equal Interval Width/Uniform Binning

This method relies on sorting the data and dividing the data values into equally spaced bin ranges. A seed \( k \) supplied by the user determines how many bins are required. With this seed \( k \), it is just a matter of finding the maximum and minimum values to derive the range and then partition the data into \( k \) bins.

4.4.1.2 Equal Frequency/Histogram Binning

Partitioning of data is based on allocating the same number of instances to each bin. Dividing the total number of instances \( n \) by \( k \), the number of bins supplied by the user does this.
4.4.2 Supervised Discretization

Supervised discretization on the other hand makes use of the instance/class labels during the discretization process to aid in the partitioning. Prior knowledge of the class label of each instance is incorporated at each iteration to refine partition breakpoint estimation of each bin.

4.4.2.1 Holte’s 1R Discretizer

This method uses an error-based approach using error counts to determine when to split intervals. The attribute is sorted into ascending order and a greedy algorithm that divides the feature into bins where each contains only one instance of a particular class label. The danger inherent in such a technique is that each instance may end up belonging to a separate bin. To combat this problem, a minimum number of instances of a particular class for each bin as specified by the user (except for the upper most bin) is implemented. Hence any given bin can now contain a mixture of class labels and boundaries will not be continually divided leading to overfitting. Each bucket grows i.e. the partition shifts to the right until it has at least 6 instances of a class label, and continues until the instance to be considered is not part of the majority class label. Empirical analysis suggests that a minimum bin size of 6 performs the best.

4.4.2.2 C4.5 Discretizer

C4.5 uses a bottom up approach that builds a complete tree and then prunes the number of intervals. Information gain theory is used to determine, at which threshold value the gain ratio is the greatest in order to partition the data. A divide and conquer algorithm is then successively applied to determine whether to split each partition into smaller subsets at each iteration. Every non-leaf sub tree is examined from the bottom and if the predicted error is lower if a leaf replaces the sub tree and then a leaf replaces the sub tree.

4.4.2.3 Fayyad and Irani’s Entropy Based MDL Method

This method uses a top down approach whereby multiple ranges rather than binary ranges are created to form a tree via multi-way splits of the numeric attribute at the same node to produce discrete bins. Determining the exact cut off points for intervals is based on the Minimum Description Length Principle (MDLP) that biases a simpler theory that can explain
the same body of data and favors a hypothesis which minimizes the probability of making a wrong decision assuming a uniform error cost. No pruning is applied to the grown tree.

An information entropy/uncertainty minimization heuristic is used to select the threshold boundaries by finding a single threshold that minimizes the entropy function over all possible thresholds. This entropy function is then recursively applied to both of the partitions induced. Thresholds are placed half way between the two delimiting instances. At this point the MDL stopping criterion is applied to determine when to stop the subdividing of discrete intervals.

4.4.2.4 Kononenko’s Entropy Based MDL Method

This method is virtually identical to that of Fayyad and Irani except that it includes an adjustment for when multiple attributes are to be discretized. This algorithm provides a correction for the bias the entropy measure has towards an attribute with many values [40].

4.5 ATTRIBUTE WEIGHTAGE METHODS

Recently, cluster ensembles have emerged as a technique for overcoming problems with clustering algorithms. It is well known that off-the-shelf clustering methods may discover different patterns in a given set of data. This is because each clustering algorithm has its own bias resulting from the optimization of different criteria. Furthermore, there is no ground truth against which the clustering result can be validated. Thus, no cross-validation technique can be carried out to tune input parameters involved in the clustering process. As a consequence, the user is equipped with no guidelines for choosing the proper clustering method for a given dataset. A cluster ensemble consists of different partitions. Such partitions can be obtained from multiple applications of any single algorithm with different initializations, or on various bootstrap samples of the available data, or from the application of different algorithms to the same dataset. Cluster ensembles are a solution to challenges inherent to clustering arising from its ill-posed nature: they can provide more robust and stable solutions by making use of the consensus across multiple clustering results, while averaging out emergent spurious structures that arise due to the various biases to which each participating algorithm is tuned, or to the variance induced by different data samples. An orthogonal issue related to clustering is high dimensionality [9].
High dimensional data pose a difficult challenge to the clustering process. Various clustering algorithms can handle data with low dimensionality, but as the dimensionality of the data increases, these algorithms tend to break down. In high dimensional spaces, it is highly likely that, for any given pair of points within the same cluster, there exists at least few dimensions on which the points are far apart from each other. As a consequence, distance functions that equally use all input features may not be effective. As a result, many different subspace clustering methods have been proposed [Parsons et al. 2004]. They all attempt to dodge the curse of dimensionality which affects any clustering algorithm in high dimensional spaces.

A common scenario with high dimensional data is that several clusters may exist in different subspaces comprised of different combinations of features. In many real-world problems, points in a given region of the input space may cluster along a given set of dimensions, while points located in another region may form a tight group with respect to different dimensions. Each dimension could be relevant to at least one of the clusters. Common global dimensionality reduction techniques are unable to capture such local structure of the data. Thus, a proper feature selection procedure should operate locally in input space. Local feature selection allows one to estimate to which degree features participate to the discovery of clusters. Such estimation is carried out using points within the local neighborhoods, and it allows the embedding of adaptive distance measures in different regions of the input space[9].

Data pre-processing is important for successful data mining, by making the data more amenable for the data mining process. Often, the raw data must be processed in order to make it suitable for analysis. While one objective may be there to improve data quality, other goals focus on modifying the data so that it better fits into a specified data mining technique. The attributes weighting (feature weighting) is one data pre-processing method, and it is an alternative to keeping or eliminate features in the applications of data mining techniques, such as classification and clustering algorithms. More important features are assigned a higher weight, while less important features are given a lower weight. These weights are sometimes assigned based on domain knowledge about the relative importance of features. Alternatively, they may be determined automatically. For example, the distance weighting is based on weighted Euclidean distance, where the inverse of the coefficient of multiple correlations was decided to be the weights.
K-Means algorithm is an effective algorithm to cluster data which has numerical attributes. It considers the contribution of each attribute of the samples as equal, but does not consider the different effects that different attributes may have on the clustering results. It was discussed about an attribute weights-based K-Means algorithm which considers the different effects of different attributes on the clustering results. This algorithm uses the inverse of coefficient of multiple correlations as the weight of the attribute, which reflects the contribution of each attribute to the clustering result, thereby increasing the accuracy of the clustering results, while simultaneously increasing the efficiency of the algorithm by decreasing the number of iterations.

4.5.1 Methods for attribute weightage

There has been numerous attribute weight setting methods for data mining proposed in the literature. Some frequently referenced methods are now summarised. One of the authors addressed the problem of combining multiple weighted clusters which belong to different subspaces of the input space. They leverage the diversity of the input clustering in order to generate a consensus partition that is superior to the participating ones. Since they deal with weighted clusters, their consensus function makes use of the weight vectors associated with the clusters. The experimental results show that their ensemble technique is capable of producing a partition that is as good as or better than the best individual clustering.

Other authors handle clustering as a constrained minimization of a Bregman divergence. Weight modifications rely on the local variations of the expected complete loglikelihoods. Theoretical results bring modified (weighted) versions of clustering algorithms. Another concept is a framework for integrating multiple, heterogeneous feature spaces in the K-Means clustering algorithm. Their methodology adaptively selects, in an unsupervised fashion, the relative weights assigned to various feature spaces with the objective of simultaneously attaining good separation along all the feature spaces. Using precision/recall evaluations, they empirically demonstrated that optimal feature weighting is extremely effective in locating the best clustering when compared against known ground truth classifications.
Some other papers describe about Support Vector Machines (SVM), feature weights can be determined automatically. This technique has its roots in statistical learning theory and has shown promising categorization possibilities. SVM produces classification models in which each feature is given a weight. Documents are often represented as vectors, where each attribute represents the frequency with which a particular term (word) occurs in the document. Cosine similarity really is a measure of the cosine of the angle between two document vectors. Thus, if the cosine similarity is 1, the angle between the two document vectors is 0°, and the two document vectors are the same except for magnitude (length). If the cosine similarity is 0, then the angle between the two document vectors is 90°, and they do not share any terms (words). In this way, the cosine similarity can be considered as a kind of feature weighting. More details about the cosine similarity can be found.

4.6 INTRODUCTION ABOUT DENSITY

A database is a container for data. The aim for a database is mainly storing in a way that the user can read and write data in an efficient way. An efficient way is realized in terms of securing data, read data in the most up-to-date state and write data in the best organizational way. Earlier databases were quite small and concurrency or performance issues were rather small. Now this is the opposite. There are huge issues and often call DBAs to the rescue for that matter. PCI or other Audit compliance adds to the mix the security of the data, that was not a main issue in the earlier days as well. Obviously there are many issues a DBA, one main issue is:

Density: Density of data is the same as people density. So often, public services have difficulties to keep Post offices open in rural areas just because the need for it is small or scarce. Government would rather manage medium and large cities. It is easier to deliver services, easier to clean streets, easier to deal with things. Of course, public services need to be organized but they gain a nice service for the buck spent.

As far as database is considered, one can say the same thing. If the data map in the databases are going everywhere on the physical disk, would have harder time to manage to clean the data and all. The thing is about having data all over in a physical or logical storage space. For example, a table would gather information about orders but the orders would be all over the place with data split everywhere.
The first thing should be known is how to index ones data if orders coming is not known. Fortunately, it would probably index the order by an identifier and then index by the date and the user that ordered mainly. But let it to be said for the sake of the example that it would not know, the orders would be all over the data map of the data. In other words, unmanageable.

The second thing is to store large empty space because the data splits in all indexes. Storage would be fragmented and one would need more of them.

The third thing is to use a lot of resources just for index maintenances when DML operations or reindexing.

The fourth thing is to decrease performance regularly because of the instability of indexes and the cost of maintenance. The concurrency locks would be longer, the transactions would take longer to perform, etc…

But the common issues to all is Density. Simply the density of data can be defined as, Data density, is how much data can fit in the same physical block of storage space. Architects would use, what is the depth of the indexes, how many data splits it performs, what is the chronic fragmentation, etc….And go on and on with different terms. They are all exact and to be a good DBA/Architect, it need to know all the concepts because it is usually called when the damage is done. Try to put as much data in the smallest storage space.

In an index/table, fill factors and covering indexes, etc… may be considered. But all of those are just strategy to minimize in the long run the same concept: Density.

The goal for covering indexes is to limit the number of indexes to create on a table so that it takes less space and then DML operations are overall smarter and reads smarter than if it create several indexes instead of the covering one.

DENSITY is the key in DBA. There are exceptions of course, otherwise the work of a DB architect and DBA would be boring. Also Front end Ecommerce databases are not into the mix. It is obviously because the main concern is not storage. It is performance and lock concurrency with volatile database. So the No Sql deal is the best one. The column stores the indexes in some cases and denormalization is some other cases is the best option, especially when working with large cache stores.

First principle for a good production database is to get only the data that is needed in the table or partition one cares. It has seen so many client databases will the entire
history of orders or transactions on production database. This fits into the principle. Obviously, it if has issues of density in a large table, density issue is smaller when the table is smaller.

The second principle is to make sure that table does not have hundreds of columns or large empty “large” data-typed columns, such as Text, Varchar (2000), xml… In other words when the data is written or read on such tables, key lookups and index updates get faster to be done. Usually developers think: why care about data types, why not putting an int instead of smallint. This is where the architect/DBA comes in. Obviously a Key-value type table would be pretty dense as there would be not much columns (assuming the clustered index would follow roughly the order where the data are inserted).

There are set-based programmings where the database is very useful, like Rating and Billing of transactions or direct marketing for example, etc… There are transactions to deal with on the diverse DML operations to set up. There are set-based programming in which security compliance security has to be done. The database engine’s assist them in that (on logins, permissions, database, backup and data encryption). But overall, main important thing is DENSITY.

There are many ways to improve density and the consequences of improving density are usually quick on performance and storage.

**4.7 DENSITY OF THE CLUSTER**

Cluster analysis is a primary method for database mining. It is either used as a stand-alone tool to get insight into the distribution of a data set, e.g. to focus further analysis and data processing, or as a preprocessing step for other algorithms operating on the detected clusters. Density-based approaches apply a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). These regions may have an arbitrary shape and the points inside a region may be arbitrarily distributed.
The idea behind constructing clusters based on the density properties of the database is derived from a human natural clustering approach. By looking at the two-dimensional database, one can almost immediately identify three clusters along with several points of noise. The clusters and consequently the classes are easily and readily identifiable because they have an increased density with respect to the points they possess. On the other hand, the single points scattered around the database are outliers, which means they do not belong to any cluster as a result of being in an area with relatively low concentration.

A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density is used when the clusters are irregular or intertwined and when noise and outliers are present. Clusters are formed by connecting neighboring 'core’ objects and those 'non-core’ objects either serve as the boundaries of clusters or become outliers. Since the noises of the data set are typically randomly distributed, the density within a cluster should be significantly higher than that of the noises. Therefore, density-based approaches have the advantage of extracting clusters from a highly noisy environment.

Density = number of points within a specified radius (Eps)

4.7.1 Selecting the cluster to split

The problem of selecting the cluster to split in divisive clustering techniques has received so far much less attention than what it deserves, since it may have a remarkable impact on the overall clustering results.

Selecting the cluster to split:

The following three approaches are typically used for the selection of the cluster to split:

1. **complete partition**: every cluster is split to obtaining a complete binary tree.
2. the cluster having the **largest number of elements** is split.
3. the cluster with the **highest variance** with respect to its centroid.

\[
\text{a (M) = } \frac{1}{N} \sum_{j=1}^{N} \left( x_j - w \right)^2 \quad \text{(Equ. 4.4)}
\]

is split (in equation 4.4 \( w \) is the centroid of data-matrix of the cluster, \( x_j \) its \( j \)-th column, is the Euclidean norm)[52].
The above criteria are extremely simple and raw. Criterion (1) is indeed a "non-choice", since every cluster is split: it has the advantage of providing a complete tree, but it completely ignores the issue of the quality of the clusters. Criterion (2) is also very simple: it does not provide a complete tree, but it has the advantage of yielding a “balanced” tree, namely a tree where the leaves are (approximately) of the same size. Criterion (3) is the most “sophisticated”, since it is based upon a simple but meaningful property (the "scatter") of a cluster. This is the reason why (3) is the most commonly used criterion for cluster selection.

The other feature, which has been introduced in the modified bisecting K-Means algorithm, is the selection of cluster for splitting further. Here, the selection of the cluster for further splitting is based on the density of the clusters. As in the definition of the bisecting K-Means algorithm, there will be two clusters generated from the first step and the procedure continues up to the closing criteria (desired number of cluster).

In normal Bisecting K-Means, the portion selected for bisecting is based on the number of data point in the bisected cluster. In the proposed algorithm, it has come up with a new strategy that is selecting the lower density area for bisecting. In detail, the low density area is selected instead of the maximum number of data points because the high density area has lots of concentrated data and in lower density the data are scattered over. In order to concentrate more on highly specific data points, select the small data concentrations in the low density area are selected as the successive points to the bisecting algorithm. The density of the cluster can be calculated using the following expression,

\[ \text{Density, } D(c) = \frac{N_c}{R_c} \]  (Equ 4.5)

In Equation 4.5 \( N_c \) is the number of data objects in cluster \( C \), \( R_c \) is the radius of cluster that is the distance of two furthest data points.

The density values will be according to the concentration about the data in the two sections. In normal bisecting K-Means algorithm, the portion labeled as one with high density is eventually selected, but in the proposed algorithm the section two is selected instead, which has low density. The section two is selected by providing a threshold for the density value. If
the value is less than the threshold that section is selected as the next section to be selected for the K-Means algorithm. The modified bisecting K-Means algorithm identifies the data concentration in the section according to its definition.

4.8 CONCLUSION:

In this chapter new method for centroid calculation is discussed in which centroid selection method is used to generate the initial cluster centers rather than random or user specified cluster centers. It is also noticed that the proposed method takes less time in execution as compared to most of the algorithms. Then regarding attribute, the description is about what is meant by attribute wherein is an attribute describes the facts, details or characteristics of an entity which is a single data item related to a database object. Different types of attributes, attributes by the number of values, attribute techniques were also discussed. Using the proposed attribute weightage method, the attribute weightage is calculated and that value is passed to the Euclidean distance for clustering. The other feature, which is introduced in the modified bisecting K-Means algorithm, is the selection of cluster for splitting further. Here, the selection of the cluster for further splitting is based on the density of the clusters. As in the definition of the bisecting K-Means algorithm, there will be two clusters generated from the first step and the procedure continues up to the closing criteria (desired number of cluster). In the next chapter the proposed method is performed step by step, that is centroid selection by new a method and then attribute weightage is calculated and the result is passed to the Euclidean distance for clustering and then the density is found out for further splitting.