Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 3, March 2016, pg.682 – 687

CLUSTERING BIG DATA USING NORMALIZATION BASED k-MEANS ALGORITHM

K.Gaja Lakshmi[1], Dr. D.Prabha[2]

[1]PG Scholar, Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India

[2]Associate Professor, Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India

[1] gajalakshmi193@gmail.com; [2] prabha@skcet.ac.in

Abstract— ***Clustering has become more challenging as the size of the data increases. In order to enhance the speed of clustering in large data sets MapReduce framework is used. MapReduce is a programming model for distributed parallel computing. An optimized k-means algorithm is used in this framework to reduce the iteration dependency which is one of the bottle necks in k-means algorithm. In k-means the initial cluster centers are chosen randomly as a result the final clusters that are obtained varies each time. In the proposed system normalization based k-means clustering algorithm is used to find the better initial centroids. By using Min-Max normalization technique initialization of centroids are done effectively. In this method weight is assigned to each attribute. So enhancement for particular feature in the dataset can be done by increasing the weight of that attribute. It also provides an efficient way to assign the data points to suitable clusters.***

Keywords— ***Clustering, MapReduce, k-means, Normalization***

I. INTRODUCTION

A new generation of technology that is designed to economically extract value from huge volumes of a variety data is referred as big data. It becomes very difficult to perform effective analysis using the existing traditional computing techniques because of its larger size. The key factors for the growth of big data are increase in storage capacities, increase of processing power and accessibility of data. To attain value from big data, an organized set of solutions for capturing, processing, analyzing the data and discovering new insights is needed. Accuracy in big data may further direct to confident decision making and accurate decisions can lead to greater operational efficiency. The availability of big data, low-cost commodity hardware and analytic software has produced a distinctive moment in the account of data analysis. It is one of the best solutions for our day to day problems such as large-scale extract, transform, and load issues by using commodity software and hardware. But still it makes certain operations like analytical, process and retrieval time intense and difficult. Clustering helps to overcome this problem by compacting it in a format that gives an informative version of the entire data. Cluster analysis has seen a vigorous growth in recent years. There are numerous types of clustering which includes partition, hierarchical, model-based, density based and grid based clustering. A good clustering method will produce high quality clusters with high intra-class similarity and low inter-class similarity. Partitioning is a basic operation in cluster analysis.

The k-means algorithm is best suitable for implementing this operation because of its effectiveness in clustering huge data sets. In this system, the focus is done on improving the efficiency of k-means to obtain a high performance. The k-means algorithm starts with the random selection of initial centers. The data points are assigned to its nearest center. This is done by computing the distance between the cluster center and the data point. Then it recomputed the center by calculating the mean and continues until the obtained result does not change for two consecutive rounds. One of the main problems in k-means algorithm is iteration dependency which depends on choosing initial cluster centers. In order to eliminate the iteration dependency a programming model called MapReduce is used [1].

The MapReduce framework helps to parallelize large computations easily by using map and reduce operations. Mainly two functions have to be specified in the MapReduce framework the map and reduce function that implements the mapper and the reducer [2]. The quality of the ensuing clusters considerably relies on the choice of initial centroids. Improper initialization can result in empty clusters, slower convergence, and a higher chance of being trapped in bad local minima.Inorder to solve this problem a normalization based k-means approach using Min-Max is being proposed. The advantage of Min-Max normalization is that preserves the relationship among the original data values. It also provides a simple way to compare values that are measured using different units of measure or scale. The calculations of initial centroids are done based on weighted average score of dataset. It is a process used to standardize all the attributes of the dataset and provide them equal weight so that redundant or noisy objects can be eliminated and there is valid and consistent data which enhances the accuracy of the result.

II. RELATED WORK

To increase the speed of large datasets and make efficient use of machine resources Jeffery [3] proposed a model in which most of the computations involved applying a map function to each logical record of the input in order to figure a set of intermediate key/value pairs. Reduce operation is then applied to all the values that shared the same key in order to combine the resulting data appropriately. The usage of functional model with user-specified map and reduce operations helps to parallelize large computations easily and to use re execution as the major mechanism for fault tolerance.

In k-means a proper initialization is essential for obtaining a good final solution. A major drawback of the k-means++ is its sequential process, which restricts its applicability to massive data: one must make k passes over the data to locate a good initial set of centers. The proposed k-means++ initialization algorithm by Bahman [4] does this by obtaining an initial set of centers that is more close to the optimum solution. It starts with a random set of k centers. In each iteration, a clustering is derived from the current set of centers. The centroids of these derived clusters then become the centers for the next iteration. The iteration is then repeated until a stable set of centers is obtained. The iterative portion of the above method is called Lloyd's iteration. In this work it shows how to drastically decrease the number of passes needed to attain, in parallel, a good initialization. The proposed initialization algorithm k-means obtains almost a finest solution after a logarithmic number of passes, and then shows a constant number of passes success.

The k-means clustering is time consuming as it converges to a local optimum of its loss function and the solution converged to be is mainly sensitive to the initial starting positions. Ian [5] proposed an approach called bootstrap averaging. It builds many models by creating small bootstrap samples of the training set and building a single model from each k-means runs only once on each sample as only single model is built. Identical cluster centers are then averaged to produce a single model that contains k clusters. The approach involves averaging similar cluster centroids by the position of attributes and their values to create signature. Grouping of clusters are done based on the signature. Bootstrap averaging on a portion of the dataset yields accurate results as clustering the entire data set. The results indicate that the size of data is directly proportional to the number of iterations of algorithm until convergence. As the size of the bootstrap sample is increased, the accuracy improves, until at some point the accuracy is similar to that when the complete dataset is used.

The optimization proposed by Cosmin[6] is based on the observation that only a small part of dataset change its cluster after performing few iterations. The implementation traces a boundary between the part of the data set which would possibly switch to another cluster and which will hold the cluster it belongs to, during the next iteration. The more iterations are performed, less centroids depart from their current position, ensuing in less data objects to be checked against.

The algorithm of Bradley [7] is intended to increase the scalability of k-means clustering for large datasets. The main idea is to use a buffer where points from the dataset are saved in compressed form. Buffer contents are compressed in two steps. The primary compression finds and discards points that are unlikely ever to move to a different cluster. On the left over points in the buffer, k-means clustering is performed, with a larger number of clusters than for the main clustering. This part is called secondary compression. After compression phases the space in the buffer is filled with new points and the entire procedure is repeated. The algorithm ends after one scan of the dataset, or if the centers of the main clusters do not vary significantly as more points are added.

A new method is proposed by Madhu[8] for finding better initial centroid with reduced time complexity. If the data set contains negative value attributes then transform those data points in the data set to the positive space by subtracting the each data point with the minimum attribute. The distance is calculated from source to each data point in the data set. So, for different data points we will get the similar euclidean distance from the source. This may outcome incorrect selection of the initial centroids. To overcome this problem all the data points are changed to positive space. If data set contains all positive value attributes then the transformation is not necessary. In next stage data points are assigned to the clusters that have closest centroids. Next, mean of the data points is taken for each cluster. Then for every data point distance is calculated from the new centroid of its current nearest cluster. The reassigning process is repeated until the convergence criterion is met.

III. K-MEANS USING MAPREDUCE

The input dataset that has to be processed is done in MapReduce framework. The MapReduce framework is a programming model for data-intensive computing [2]. The framework based on hadoop requires a pair of map and reduces functions. The map and reduce function are used for processing and generating large data sets. The collected dataset is divided into equal partitions and the framework assigns one partition to each map function. In this phase, each mapper reads the input data and the selects the random center. It iterates over each cluster centroid for its input data points. The euclidean distance is calculated and the data points are assigned to the clusters which has its lowest distance.

After every map task, a combiner is applied to combine the intermediate data of the same map task. In order to compute the mean value of the objects for every cluster, the record is done on the number of samples in the same cluster and in same map task. The MapReduce framework guarantees the input to every reducer to be arranged by key. The procedure by which the system sorts and passes map output to reducers is known as shuffle. In reduce function, sum every sample and calculate the total number of samples assigned to the same cluster. The new centers are obtained which are used for next iteration till the convergence criteria are met.

IV. PROPOSED SYSTEM

Normalization based k-means clustering algorithm applies normalization on the available data prior to clustering as well as the proposed method calculates initial centroids based on weights. Normalization is used to eradicate redundant data and it improves the efficiency of clustering algorithms. So it becomes an important step prior to clustering as euclidean distance is very sensitive to the changes in the differences.

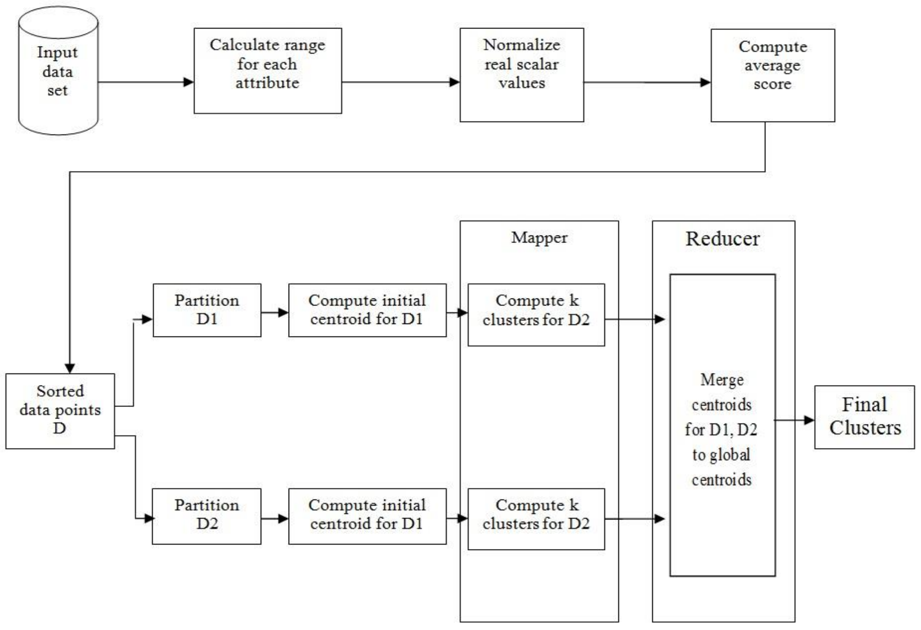

Input data set
Calculate range for each attribute
Normalize real scalar values
Compute average score
Sorted data points D
Partition D1
Partition D2
Compute initial centroid for D1
Compute initial centroid for D1
Mapper
Compute k clusters for D2
Compute k clusters for D2
Reducer
Merge centroids for D1, D2 to global centroids
Final Clusters

Fig 1: Architecture of normalization based k-means

The Max-min distance measure has significant advantage over euclidean distance, because it takes less number of iterations to attain convergence. k-means algorithm can produce better results after the modification of the databases. The normalization process involves three phases. The first stage involves weighted average score and initial centroids calculation. The second stage involves implementing map function in k-means and then implementing the reduce function

A. *Weighted average score and initial centroids calculation*

Normalization is used to standardize all the attributes in the dataset. Equal weights are assigned so that redundant data are eliminated. In this process Min-Max normalization technique is used which performs linear transformation on the data. From the loaded input dataset the minimum and maximum values are taken. Suppose min $_e$ and max $_e$ are minimum and maximum for attribute E. Min-max normalization maps a value v of E-v in the range (0, 1) by calculating

$$v' = \frac{v - \min(e)}{\max(e) - \min(e)}$$

To calculate the average score,
Di (avg) = (w1*x1+w2*x2+w3*x3+…wm*xm)/m
where, x=attribute's value, m=no of attributes, w=weight to ensure uniform distribution of clusters
The calculated average score is sorted using sorting algorithm. The sorted data points are then divided into k-subsets where k is the number of clusters. The mean of each subset is calculated. The nearest possible value of mean from each subset is taken as initial centroid. In ordinary k-means algorithm, the initial centroids are randomly computed. As a result, the cluster centroids are recalculated several times before the convergence criteria of the algorithm are met and the data points are assigned to suitable nearest centroids.

B. *Implementation of map function*

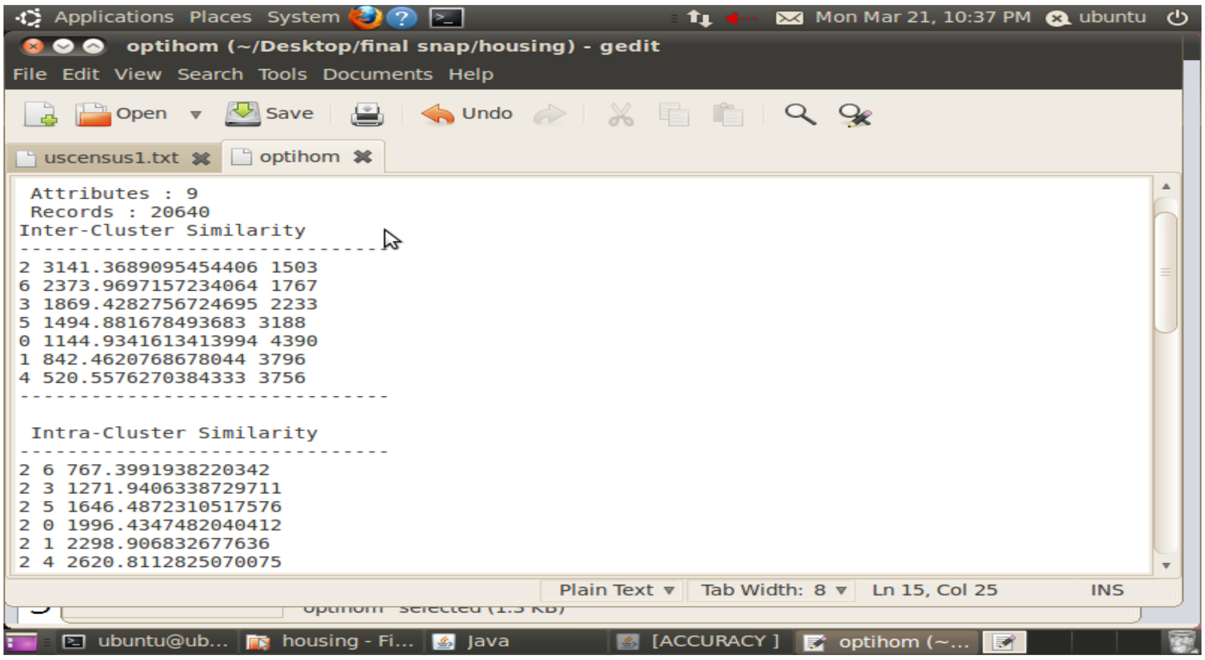
The initial centroids and the dataset are read on each mapper using setup function. In map function the user is required to handle the input of a pair of key value and produces a set of intermediate key and value pairs. It then combines the intermediate values with same key and passes them to reduce function. For every Input Split the MapReduce framework generates a map task, and it is generated by the InputFormat of job. Each <key, value> correspond to a map task. For every map task, it has centers which is an array carrying the information about centers of the clusters. A mapper can compute the closest center point for each sample with the given information. Iterate over each cluster centroid for each input key/value pair. Compute the euclidean distance and save the nearest cluster with the lowest distance to the input key/value pair. Perform the map task the input <key, value> will be processed to form a new <key, value>. This method is called "divide into groups". The intermediate values consist of two parts: the index of the closest center point and the sample information.

C. *Implementation of reduce function*

Each map function output is allocated to a particular reducer by the application's partition function. The reducer receives a key/ value pair where key is the centroid and value is the list of all data points assigned to that cluster. The framework calls the applications reduce function once for each unique key in the sorted order. It is important to pick a partition function that gives an approximately uniform distribution of data otherwise the MapReduce operation can be held up waiting for slow reducers to finish. The list is iterated to obtain the average data point. This gives set of new centroids. This is repeated for every key, value pairs on the reducer. Finally, the centroids produced are compared to the centroids produced in the previous step to check if convergence criteria are met. If the centroids have not converged, the module is iterated with the new centroids until met. The centroids are merged to obtain global centroids and the final clusters are obtained.

V. EXPERIMENTAL DATASET

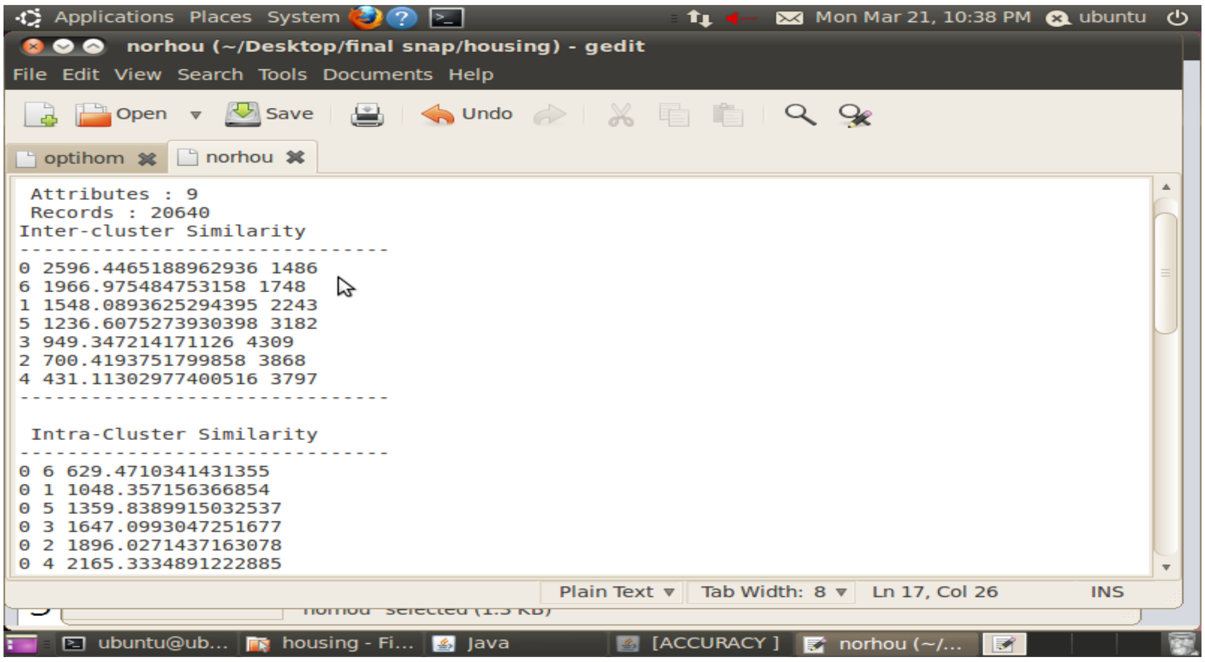
The implementation is tested in the housing dataset which has nine attributes namely Longitude, latitude, housing median age, total rooms, total rooms occupied, population, households median income, median house value. The dataset contains 20,700 instances of different attribute values. The dataset is processed in k-means using MapReduce and normalization based k-means algorithm. Fig 2 and 3 shows output for k-means using MapReduce and normalization based k-means.


Applications Places System
Mon Mar 21, 10:37 PM ubuntu
optihom (~/Desktop/final snap/housing) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
uscensus1.txt
optihom
Plain Text
Tab Width: 8
Ln 15, Col 25
INS
ubuntu@ub...
housing - Fi...
Java
[ACCURACY]
optihom (~...

```
 Attributes : 9
 Records : 20640
Inter-Cluster Similarity
-------------------------------
2 3141.3689095454406 1503
6 2373.9697157234064 1767
3 1869.4282756724695 2233
5 1494.881678493683 3188
0 1144.9341613413994 4390
1 842.4620768678044 3796
4 520.5576270384333 3756
-------------------------------

 Intra-Cluster Similarity
-------------------------------
2 6 767.3991938220342
2 3 1271.9406338729711
2 5 1646.4872310517576
2 0 1996.4347482040412
2 1 2298.906832677636
2 4 2620.8112825070075
```

Fig 2 Output of k-means using MapReduce


Applications Places System
Mon Mar 21, 10:38 PM ubuntu
norhou (~/Desktop/final snap/housing) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
optihom
norhou
Plain Text
Tab Width: 8
Ln 17, Col 26
INS
ubuntu@ub...
housing - Fi...
Java
[ACCURACY]
norhou (~/...

```
 Attributes : 9
 Records : 20640
Inter-cluster Similarity
-------------------------------
0 2596.4465188962936 1486
6 1966.975484753158 1748
1 1548.0893625294395 2243
5 1236.6075273930398 3182
3 949.347214171126 4309
2 700.4193751799858 3868
4 431.11302977400516 3797
-------------------------------

 Intra-Cluster Similarity
-------------------------------
0 6 629.4710341431355
0 1 1048.357156366854
0 5 1359.8389915032537
0 3 1647.0993047251677
0 2 1896.0271437163078
0 4 2165.3334891222885
```

Fig 3: Output of normalization based k-means

The results are also verified with the other datasets in UCI repository. The results show that normalization based k-means is better than k-means using map reduce while comparing it with the similarity measure. It shows proper assigning of data points to the suitable cluster. The graph below in fig 4 shows the comparison graph

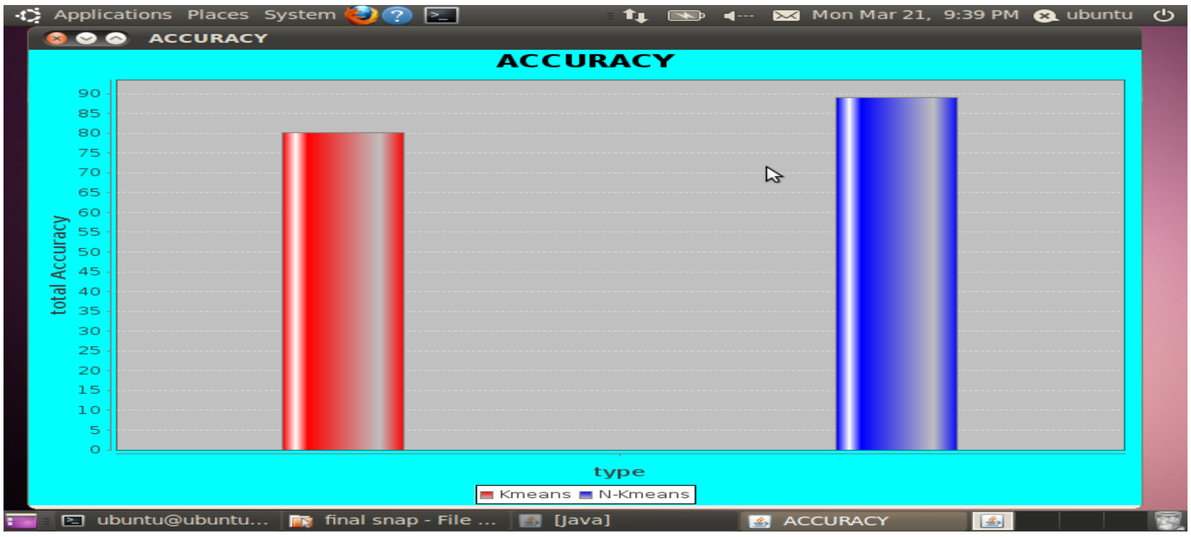


Fig 4: Comparison Graph

The comparison graph of k-means using MapReduce and normalization based k-means shows that normalization based algorithm is better.

VI. CONCLUSION

The normalization based k-means algorithm improves the initial center points and provides an efficient way of assigning data points to suitable clusters. Iteration dependency one of the biggest bottle necks of k-means is also eliminated by using MapReduce framework. When normalization approach is used the results show that the appropriate clusters are obtained with high intra-cluster similarity and low inter-cluster similarity.

REFERENCES

[1] Xiaoli Cui, Pingfei Zhu, Xin Yang, Keqiu Li Changqing Ji, "Optimized big data k-means clustering using MapReduce", Journal of Supercomputing, Springer, Volume 70, pp: 1241-1259
[2] ZhaoW, Ma H, He Q, "Parallel k-means clustering based on MapReduce", Cloud computing, Springer, pp: 674-679, 20
[3] Dean J, Ghemawat S, "MapReduce: simplified data processing on large Clusters", Communications of the ACM, pp: 107–113, 2008
[4] Bahmani B, Moseley B, Vattani A et al, "Scalable k-means++", Journal Proceedings of Very Large Data Bases Endowment, pp: 622–633, 20125
[5] Davidson I,Satyanarayana A, "Speeding k-means clustering by bootstrap averaging", IEEE data mining workshop on clustering large dataets,pp:49-60,2003
[6] Cosmin Marian Poteras, Marian Cristian Mihaescu, Mihai Mocanu "An optimized version of the k-means clustering algorithm", Federated Conference on Computer Science and Information Systems, pp: 695–699, 2014
[7] Farnstrom F, Lewis J, Elkan C, "Scalability for clustering algorithms Revisited", ACM SIGKDD Explore News, pp:51–57, 2000
[8] Mugdha Jain, Chakradhar Verma, "Adapting k-means for clustering in big data",International Journal of Computer Applications, pp: 19-24, 2014