Density-based and partitioning-based clustering on large threshold-bounded data sets

by

ARUNA SAI KANNAMAREDDY

B.Tech., Jawaharlal Nehru Technological University, 2013

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2017

Approved by:

Major Professor
Dr.William H.Hsu

Abstract

The project explores the possibility of increasing efficiency in the clusters formed out of massive data sets, which are formed using the threshold blocking algorithm, clusters thus formed are denser and qualitative. Clusters that are formed out of individual clustering algorithms alone, do not necessarily eliminate outliers and the clusters generated can be complex, or improperly distributed over the data set. The threshold blocking algorithm, a current research paper from Michael Higgins of Statistics Department on other hand, in comparison with existing algorithms performs better in forming the dense and distinctive units with predefined threshold. Developing a hybridized algorithm by implementing the existing clustering algorithms to re-cluster these units thus formed is part of this project.

Clustering on the seeds thus formed from threshold blocking algorithm, eases the task of clustering to the existing algorithm by eliminating the overhead of worrying about the outliers. Also, the clusters thus generated are more representative of the whole. Also, since the threshold blocking algorithm is proven to be fast and efficient, we now can predict a lot more decisions from large data sets in less time. Predicting the similar songs from million-song dataset using such a hybridized algorithm is considered as the data set for the evaluation of this goal.

Table of Contents

List of Figures

Acknowledgements

I am fortunate enough to be part of the team that is working closely with Dr. Michael Higgins from the KSU Statistics department and Dr. William Hsu from the KSU Computer Science department on a research experiment. I would like to thank Dr. William Hsu for his constant inputs, feedback and guidance in delivering this project. I would also like to thank Dr. Michael Higgins for his constant support in the domain of expertise, guidance, feedback and for taking the initiative to collaborate with us. I would also like to thank Dr. Mitch Neilsen and Dr. Torben Amtoft for serving on my M.S. committee. Finally, I would like to thank my family and friends for their love and support.

Chapter 1 - Introduction

(Higgins, Savje, & Sekhon, 2016) describes a novel approach for sampling algorithm to any covariate dataset. The goal of this project aims at analyzing the performance variations of the mentioned threshold blocking algorithm in association with prominent clustering algorithms. In order to achieve this goal, the following sequence is carried out.

The threshold blocking algorithm is used along with clustering algorithm to form clusters among the dataset. Since blocks formed out of threshold blocking algorithm contain centroids representing similar units, clustering algorithm applied on the centroids of these blocks form more efficient clusters. For this purpose, DBSCAN (**D** ensity-**B** ased **S** patial **C** lustering of **A** pplications with **N** oise) and k-means clustering algorithms are used in this project. The clusters formed from DBSCAN in association with threshold blocking algorithm are compared against the clusters formed from running DBSCAN alone on the data. In the similar way, the clusters formed from the k-means clustering in association with threshold blocking algorithm are compared against the clusters formed from running only k-means clustering on the data. Thus, the performance and compatibility of the algorithm with prominent clustering algorithms is analyzed.

Chapter 2 - Background

Experiments executed with random samples chosen from the data ensure that estimated treatment effects are equal to the true causal effects of interest in expectation. However, the assigned data sample may not be a right fit to test the experiment result or effect. For Example, consider a medical study on the effect that a drug has on life expectancy, it may occur by chance that the control group is older and sicker than the treatment group. In such cases, there is high likelihood to observe inaccurate estimations or results as there are imbalances in covariates. Therefore, the studies based on such data contain high variance and the results from the data tend to be biased conditionally on the distribution of covariates.

Unadjusted estimates for even massive experiments are often too variable to enable reliable inferences because the effects of interest may be small and distributional issues result in surprisingly large variances. In the case of massive data, the experiment of interest might be draw fine-grained inferences and targeting the treatments to subgroups. Due to dimensionality curse and random assignment, subgroups of interest used for such experiments might lack sufficient data needed for analysis.

Blocking has become the default experimental design of choice for dealing with the above scenarios. With this design, the investigator forms groups of units, or blocks, that are as similar as possible. Treatments are then randomly assigned in fixed proportions within blocks and independently across them. This prevents imbalances in observed covariates, which can increase precision if these covariates are predictive of outcomes. Blocking improves precision in the test result by adjusting for covariates in the design of study rather than from the test result.

Also, existing blocking methods are not sensitive to clustering of data points and are often heuristic. Therefore, the samples generated by these blocking methods do not form a good dataset to the clustering algorithms and thereby leading to erroneous results. Also, the existing algorithms that are proven to be optimal are computationally expensive and especially not feasible for large data sets.

Considering all the above scenarios, the proposed threshold blocking algorithm aims to solve all these problems. The algorithm takes an input to threshold value, which is minimum number of points to be contained in each block or group and a distance metric. The algorithm tries to minimize the maximum distance between any two units in the same group. Thus, the algorithm offers flexibility in the block structure and forms blocks resembling natural cluster units, which may improve performance. One more advantage of threshold blocking algorithm when compared to fixed size blocking is that in the case of fixed size we might not respect natural clustering of units and one is sometimes forced to assign similar units to different blocks just to satisfy the cardinality condition where as in the threshold blocking we can specify the number of units a cluster should contain based on the type of data thus respecting natural clustering of units.

2.1 NP-Hardness of Threshold Blocking Problem

We consider the blocking problem where one wants to minimize the greatest within-block dissimilarity, as measured by an arbitrary distance metric, subject to a minimum required block size. Solving this is an NP-Hard problem. Let us see why this is an NP Hard Problem.

Let k denote a threshold for the minimum block size. Consider the complete graph G = (V, E) describing an experimental sample, where V denotes the set of n vertices (the experimental units) and E denotes the set of edges connecting all pairs of vertices. For each $ij \in E$ there is an associated cost, c_{ij} , indicating the dissimilarity between i and j; lower costs mean that units are more similar. We require that these costs satisfy the triangle inequality:

$$\forall ij, jl, il \in E , c_{ij} + c_{jl} \geq c_{il} \qquad (1)$$

This ensures that the direct route between two vertices is no longer than a detour through a third vertex. All distance metrics fulfill this criterion by definition.

Definition 1: A threshold blocking with threshold k is a partition $b = \{V_1 \ldots V_m\}$ of V where each block satisfies the size threshold:

$$\forall V_x \in b, |V_x| \geq k \qquad (2)$$

Definition 2: The subgraph generated by a blocking $b = \{V_1 \ldots V_m\}$, denoted G(b) = (V,E(b)), is the union of subgraphs of G induced by the components of b; that is, an edge $ij \in E(b)$ only if i and j are in the same block:

$$E(b) \equiv \{ij \in E: \exists V_x \in b, i, j \in V_x\} \tag{3}$$

Let B_k denote the set of all possible threshold blockings of G with a threshold of k. The bottleneck threshold blocking problem is to find a blocking in Bk such that the
maximum within-block dissimilarity is minimized. This amounts to finding an optimal blocking $b^* \in B_k$ such that the largest edge cost in $G(b^*)$, is as small as possible; let
λ denote this minimum:

$$\max_{ij \in E(b^*)} c_{ij} = \min_{b \in B_k} \max_{ij \in E(b)} c_{ij} \equiv \lambda \tag{4}$$

Definition 3: An α-approximation algorithm for the bottleneck threshold blocking problem derives a blocking $b \in B_k$ with a maximum within-block cost no larger than $\alpha\lambda$:

$$\max_{ij \in E(b)} c_{ij} \leq \alpha \lambda \tag{5}$$

So unless P = NP, no polynomial-time $(2 - \epsilon)$ −approximation algorithm exists for any $\epsilon > 0$. Therefore, the problem is NP-hard, and finding an optimal solution is computationally intractable except for special cases or very small samples.

2.2 Approximate of Threshold Blocking Algorithm

The threshold blocking problem can be solved with 4-approximation algorithm. The algorithm guarantees a threshold blocking with maximum within block no longer than 4λ.

$$\max_{ij \in E(b_{alg})} c_{ij} \leq 4\lambda$$

Proof: Before going into proof let's look at our lemma's

Lemma 1: For any non-seed vertex, $i \notin S$:

1. There exist no two seeds both adjacent to i in G_{nn}.
2. There exists a walk in G_{nn} of two or fewer edges from i to the seed of the block that i is assigned to.

Lemma 2: No edge cost in G_{nn} can be greater than the maximum cost in the optimal blocking

Let b_{alg} denote the blocking produced by the algorithm. Consider any within-block edge $ij \in E(b_{alg})$.We must show that c_{ij} is bounded by 4λ.

If $ij \in E_{nn}$, we have $c_{ij} \leq \lambda$ by Lemma 2. If $ij \notin E_{nn}$ and $i \notin S, j \in S$, then by Lemma 1, there exists some l so that $il, lj \in E_{nn}$. Lemma 2 applies to both these edges. By Equation 1, the triangle inequality, it follows:

$$c_{ij} \leq c_{il} + c_{lj} \leq \lambda + \lambda = 2\lambda$$

If $ij \notin E_{nn}$and $i, j \notin S$, let $l \in S$ be the seed in the block that vertices i and j are assigned to. From above we have $c_{il} + c_{lj} \leq 2\lambda$, and by the triangle inequality:

$$c_{ij} \leq c_{il} + c_{lj} \leq 2\lambda + 2\lambda = 4\lambda$$

As there is exactly one seed in each block, $i, j \in S$ is not possible and we have considered all edges in $E(b_{alg})$.

2.3 Algorithm

Given the graph representation of the experimental sample, G = (V, E), and a pre-specified threshold k, the approximate blocking algorithm proceeds as follows:

Algorithm1: Threshold Blocking

```
1 Construct a (k-1)-nearest neighbor subgraph of G. Denote this graph
  $G_{nn} = (V, E_{nn})$.
2 Find a maximal independent set of vertices, S, in the second power of the (k-1)-nearest
  neighbor subgraph, $G^2_{nn}$. Vertices in S are referred to as the block seeds.
3 For each seed i S, create a block comprised of its closed neighborhood in
  $G_{nn}, V_i = N_{G_{nn}}[i]$
4 For each yet unassigned vertex, assign it to any block that contains one of its adjacent vertices
  in $G_{nn}$.
```

When the algorithm terminates, the collection of blocks, $b_{alg} = \{V_i\}_{i \in S}$, is a valid threshold blocking of the experimental units that satisfies the optimality bound.

2.4 Complexity

The blocking algorithm terminates in T(n) = $O(n \log^k n)$ using $O(kn)$ space. Currently, available or any of the commonly used blocking algorithms run in polynomial time, but the threshold blocking algorithm runs in quasilinear time. Moreover, in the case of fixed k and an efficient nearest neighbor subgraph construction algorithm, blocking algorithm runs in $O(nlogn)$ time and O(n) space complexity.

Chapter 3 - Literature Survey

3.1 Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithm

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN) is a data clustering algorithm proposed by (Martin, Hans-Peter, Jiirg, & Xu, 1996). It is a density-based clustering algorithm. Given a set of points in some space, the algorithm groups together points that are closely packed together i.e., points with many nearby neighbors while marking as outlier's points that lie alone in low-density regions whose nearest neighbors are too far away.

The brief description of the algorithm is as follows: -

Given a set of points in some space, which are to be clustered, they are classified as core points, density reachable points and outliers.

- Core Points: - A point p is a core point if at least minPts points are within distance ε (ε is the maximum radius of the neighborhood from p) of it (including p). Those points are said to be *directly reachable* from p. By definition, no points are directly reachable from a non-core point.
- Reachable Points: - A point q is reachable from p if there is a path p_1, ..., p_n with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i (all the points on the path must be core points, with the possible exception of q).
- Outliers: - All points not reachable from any other point are outliers.

Now if p is a core point, then it forms a *cluster* together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.

3.1.1 Algorithm

Algorithm : DBSCAN

```
Initialisation: $C$ =0
foreach point $P \in D$ do
    if P is visited   then   Continue next point

    mark P as visited
    NeighborPts = regionQuery (P,eps)
    if sizeof (NeighborPts) ≤ MinPts   then   Mark P as NOISE

    else
        C = nextcluster ;
        expandCluster (P,NeighborPts,C,eps,MinPts)
```

A cluster then satisfies two properties: -

- All points within the cluster are mutually density-connected where two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are density-reachable from *o*. Density-connectedness *is* symmetric.

- If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

Parameter Estimation

Every data mining task has the problem of parameters. Every parameter influences the algorithm in specific ways. For DBSCAN, the parameters ε and *minPts* are needed. The parameters must be specified by the user. Ideally, the value of ε is given by the problem to solve (e.g. a physical distance), and *minPts* is then the desired minimum cluster size.

- *MinPts*: As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions *D* in the data set, as $minPts \geq D + 1$. The low value of $minPts = 1$ does not make sense, as then every point on its own will already be a cluster. With $minPts \leq 2$, the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height ε. Therefore, *minPts* must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. The larger the data set, the larger the value of *minPts* should be chosen.

- ε: The value for ε can then be chosen by using a k-distance graph, plotting the distance to the *k = minPts* nearest neighbor. Good values of ε are where this plot shows a strong bend: if ε is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of ε, clusters will merge and the majority of objects will be in the same cluster. In general, small values of ε are preferable, and as a rule of thumb only a small fraction of points should be within this distance of each other.
- Distance function: The choice of distance function is tightly coupled to the choice of ε, and has a major impact on the results. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter ε can be chosen.

3.1.2 Advantages

1. DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means.
2. DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
3. DBSCAN has a notion of noise, and is robust to outliers.
4. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database. (However, points sitting on the edge of two different clusters might swap cluster membership if the ordering of the points is changed, and the cluster assignment is unique only up to isomorphism.)
5. DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R* tree.
6. The parameters minPts and ε can be set by a domain expert, if the data is well understood.

3.1.3 Complexity

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the number of region query invocations mostly governs the time complexity. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes a neighborhood query in *O(log n)*, an overall average runtime complexity of *O(n log n)* is obtained (if parameter ε is chosen in a meaningful way, i.e. such that on average only *O(log n)* points are returned). Without the use of an accelerating index structure, or on degenerated data (e.g. all points within a distance less than ε), the worst case run time complexity remains O(n^2). The distance matrix of size $(n^2-n)/2$ can be materialized to avoid distance recomputations, but this needs O(n^2) memory, whereas a non-matrix based implementation of DBSCAN only needs O(*n*) memory.

3.2 K-Means Algorithm

K-Means is an unsupervised learning technique, giving us the information about underlying structure in the data without being told the labels. K-means clustering is the most popular partitioning method. It requires the analyst to specify the number of clusters to extract. A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters. A bend in the above plot gives us the number of clusters to be given as input to algorithm. This method is used if number of clusters in the dataset or the number of clusters dataset is to be K-means works by separating the training data into *k* clusters. It calculates the center point (mean) of each cluster, giving *k* means. New data points are clustered based on their distance to all the cluster centers: the nearest cluster is considered the most similar and best fit. divided is not known.

3.2.1 Algorithm

The k-means algorithm divides a set of N samples X into K disjoint clusters$C$, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids"; note that they are not, in general, points from$X$, although they live in the same space. The K-means algorithm aims to choose centroids that minimize the within-cluster sum of squared criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$$

K-means is often referred to as Lloyd's algorithm. In basic terms, the algorithm has three steps.

- The first step chooses the initial centroids, with the most basic method being to choose k samples from the dataset X. After initialization, K-means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid.
- The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid.

The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly.

3.2.2 Advantages

1. It scales well to large number of samples and has been used across a large range of application areas in many different fields.
2. As such, it has been successfully used in various topics, including market segmentation, computer vision, geo-statistics, astronomy and agriculture. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.
3. K-Means is used for vector quantization, feature learning and cluster analysis.

3.2.4 Complexity

Regarding computational complexity, finding the optimal solution to the k-means clustering problem for observations in d dimensions is:

1. NP-hard in general Euclidean space d even for 2 clusters.
2. NP-hard for a general number of clusters k even in the plane.
3. If k and d (the dimension) are fixed, the problem can be exactly solved in time O($n^{\{dk+1\}}$), where n is the number of entities to be clustered.

Chapter 4 - Dataset

The Million Song Dataset is a freely available collection of audio features and metadata for a million contemporary popular music tracks. The dataset contains only the feature analysis and metadata for one million songs but not the audio provided by (Lamere, Million Song Dataset, 2011) . The size of the entire dataset is around 280GB containing almost one million song records. The features of each record in the dataset consists of the following features.

4.1 Data Filtering

The data is distributed using hdf5 files, which are converted, to .csv extension files using python wrapper. The created .csv files are further filtered to retrieve only the required parameters for the analysis of given problem. Thus, from the .csv files the fields loudness, tempo, time_signature, duration and key are filtered to form the dataset used in the prediction task.

4.2 Data Validation

Cross-Validation is used to validate the model to check how the statistical analysis results will generalize to an independent data set. It is used here to estimate how accurately our predictive model will perform in practice. Usually in a supervised learning for a prediction problem, the known set of data (i.e., data with cluster labels) is partitioned into training data and testing data. The model is trained on training data and is validated against testing data. This is done to avoid problems like overfitting and will give an insight on how the model will generalize to an independent dataset. In case of unsupervised learning for prediction problem, the dataset does not contain labels to follow the same approach. So, the cross-validation is done against the error rate on clustering results of training data and test data. In this project, average intra-cluster distances are used to identify the differences in clustering results of training data and test data.

In the holdout method, we randomly assign data points to two sets d0 and d1, usually called the training set and the test set, respectively. The size of each of the sets is arbitrary although typically the test set is smaller than the training set. We then train on d0 and test on d1. In typical cross-validation, multiple runs are aggregated together; in contrast, the holdout method, in isolation, involves a single run. While the holdout method can be framed as "the simplest kind of cross-validation", many sources instead classify holdout as a type of simple validation, rather than a simple or degenerate form of cross-validation. We start from 90% training data and 10% test data. The training data size percentage is decreased by 10% and test data set size is increased by 10% in each iteration of hold-out cross-validation. This is carried out until we reach 10% training data and 90% test data.

Hold out cross-validation is done against the threshold values suitable for both DBSCAN and K-Means algorithm to run on entire 1 million dataset. The below process is illustrated for 1 run of cross-validation among the 10 folds. This process remains same for the rest of the folds but the size of training and test data set changes with each fold.

1. In the project, training data and test data are formed out of the samples of the dataset.
2. The hybridized algorithm model is trained on the training data and thus, formed model is used to predict the cluster number for the testing data. The process of prediction will not affect in changing the cluster center formed out of the model.
3. The inter-cluster and intra-cluster average distances for the clusters are used as measures to validate the system. These measure are used to validate the model.
4. These measures are calculated for the clusters that are formed from training data is validated against the clusters that are formed after merging the test data with training data model clusters.
5. The measures are to be similar in order to avoid overfitting of the model.

4.3 Feature Selection

The original Million Song Dataset does not contain any labels or genre information. The goal of clustering task in the project is to predict the genre label of each song. To identify the features corresponding to the task on given data set, a feature selection algorithm PCA or random forest is chosen. An alternative data set exists which contains partial data from the Million Song Dataset along with genre labels. This dataset is chosen to identify the predictors.

Chapter 5 - Proposed Model

The project aims to study the performance of threshold algorithm in association with clustering algorithms. The focus is on how the efficiency and accuracy of the clustering algorithm is bolstered by the use of threshold blocking algorithm as a preprocessing step. The threshold blocking algorithm published in the paper, when combined with either K-Means or DBSCAN for clustering purpose is referred to as hybridized clustering algorithm. In order to obtain the performance metrics such as efficiency and accuracy, the hybridized clustering algorithm is compared against the clustering algorithm. Data is run individually on both on hybridized clustering algorithm and as well as on the clustering algorithm. The clusters thus formed out of both the approaches are evaluated against parameters like inter-cluster distance, intra-cluster distance, Silhouette Coefficient and similarity between the cluster outputs. This project uses K-Means and DBSCAN as clustering algorithms to study the performance threshold blocking algorithm by carrying out the below two experimental approaches: -

- Threshold blocking algorithm with K-Means vs K-Means.
- Threshold blocking algorithm with DBSCAN vs DBSCAN.

The project uses Million Song Dataset to evaluate the performance in both the experiments. The Million Song Dataset, which is a covariate data, is chosen for this experiment and the clusters thus formed out of this data depict the similarity between the songs in the dataset. The clusters formed by running the clustering algorithms are expected to produce different genre sets, where each cluster is representative of a genre. So, all the songs in a cluster belong to a genre which is different from the genre of a point belonging to different cluster. As the dataset contains only 13 genres, we run the clustering algorithms to divide the data into 13 clusters.

Initially, The Million Song Dataset is given to threshold blocking algorithm to form clusters such that each cluster contains minimum number of elements equal to threshold value. These samples are closely connected points in multi-dimensional space. The threshold value ensures that data is divided into samples, where each sample consists of points with high similarity measure between any two points in the sample. The centroid calculated for the sample represents the characteristics of sample as a whole. The centroids calculated from each of these clusters is given to both k-means clustering algorithm and DBSCAN algorithm. So, the project consists of two parts. Firstly, we analyze the performance and validate the results of hybridized algorithm consisting of threshold blocking algorithm and k-means algorithm. Secondly, the same steps are repeated against threshold blocking algorithm and DBSCAN algorithm.

5.1 K-Means with Threshold Blocking Algorithm

The Million Song Data set is initially clustered using a random k value by threshold blocking Algorithm. The centroids of the above clusters formed out of this algorithm is given as input to K-Means Algorithm. The K-Means algorithm is made to divide these centroids into 13 clusters where each cluster representing the genre. The centroid of the sample and the points corresponding to the sample are clustered into the same cluster consisting of the centroid of the sample. Since, the centroid of the sample represents it as a whole, the points of the sample as well can be clustered into the same cluster as the centroid. Thus, all the records of the data set are divided into 13 clusters.

On the other hand, the entire data set is given to K-Means for cluster analysis. The clusters thus formed using K-Means are compared against the clusters formed by above hybridized algorithm to check how many points overlap and how many points do not overlap. Also, with various values of k , the change in intra-cluster and inter-cluster distances, time, memory and other such cluster evaluation factors are used to depict the performance of hybrid algorithm.

5.2 DBSCAN with Threshold Blocking Algorithm

In the DBSCAN, the first step of sampling based on the k value remains same as above. The Million Song Dataset in divided into samples or clusters consisting of minimum k points in each sample. The centroids of these samples are passed to DBSCAN for analysis. The minimum number of points and epsilon values that are to be given as input to DBSCAN are determined using KNNDistplot. These values and the centroids of the samples are passed to DBSCAN for cluster analysis.

DBSCAN do not require the number of clusters to be given to the algorithm. DBSCAN generates clusters of arbitrary number representing the genres. The cluster evaluation metrics like intra-cluster and inter-cluster distances, time, memory are calculated for the generated clusters. These metrics are calculated for every instance of k value that is passed to algorithm. A range of k values are chosen to be given as input to the threshold blocking algorithm like in k-means to check the performance variance over various values of k.

DBSCAN is also run on the dataset without any processing step of threshold blocking algorithm. The clusters thus generated are used to compare the similarity with the clusters generated by DBSCAN and threshold blocking algorithm. The metrics of generated clusters are also computed which are compared with the hybridized algorithm for every instance of k.

5.3 Data Flow Diagram

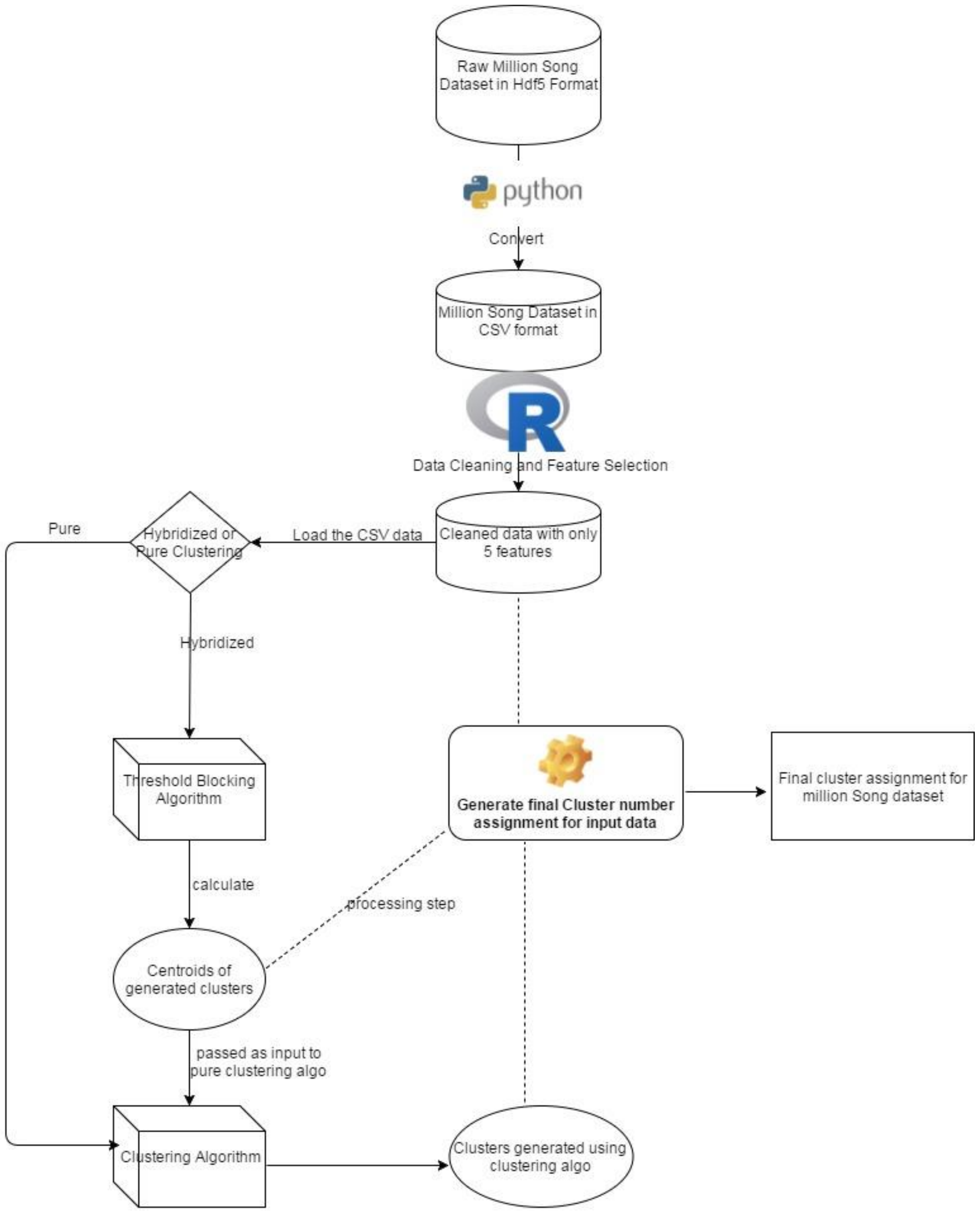
Raw Million Song Dataset in Hdf5 Format
python
Convert
Million Song Dataset in CSV format
Data Cleaning and Feature Selection
Cleaned data with only 5 features
Load the CSV data
Hybridized or Pure Clustering
Pure
Hybridized
Threshold Blocking Algorithm
calculate
Centroids of generated clusters
processing step
passed as input to pure clustering algo
Clustering Algorithm
Clusters generated using clustering algo
Generate final Cluster number assignment for input data
Final cluster assignment for million Song dataset

Figure 1 Data Flow Diagram

5.4 Implementation

1. The Million Song Dataset is obtained from the following source. (Lamere, The Million Song Dataset)
2. The files provided by the dataset are in .h5 format which are converted using python wrapper code into .csv files.
3. Data is pre-processed by removing the covariate variables and features obtained by running feature selection algorithm are retrieved either from the .csv files or during the conversion from .h5 to .csv files. This forms the dataset for the project.
4. Dataset consisting of million records is used for running the hybridized K-Means, hybridized DBSCAN, DBSCAN and K-Means algorithm.
5. Due to computation limits, entire data set for clustering is given to hybridized K-Means and K-Means algorithm while a random subset is chosen for clustering to DBSCAN and hybridized DBSCAN. It is cross-validated against the K-Means model before running the experiment.
6. R wrapper of the threshold blocking algorithm provides an implementation of the algorithm. This library is used to initially run on the dataset chosen for the experiment i.e., 1 million record dataset for K-Means and 30,000 record dataset for DBSCAN. Given a threshold value k, the algorithm divides the dataset into blocks which consist of minimum k points.
7. For each of these clusters formed out of threshold blocking algorithm, centroids are calculated such that it represents the block as a whole. These centroids are written to another file which is given as input to K-Means or DBSCAN algorithm for clustering.
8. In Experiment 1, the output from threshold blocking algorithm in given to K-Means algorithm for clustering. On the other part, the data set, which is given to threshold blocking algorithm, is given to K-Means to compute the clusters for the data set. The output from hybridized K-Means algorithm is compared against the output of K-Means algorithm to evaluate the performance of the algorithm.
9. In Experiment 2, the output from threshold blocking algorithm in given to DBSCAN algorithm for clustering. On the other part, the data set which is given to threshold blocking algorithm is given to DBSCAN to compute the clusters for the data set. The output from hybridized DBSCAN algorithm is compared against the output of DBSCAN algorithm to evaluate the

performance of the algorithm. Before running the DBSCAN algorithm, based on the knndistPlot parameter values are estimated.

10. The above two experiments are carried out for various threshold values given to threshold blocking algorithm and evaluated against various metrics.

5.5 System Configuration

Operating System: Windows 64-bit Operating System

Programming Language: R

RAM: 32 GB Memory

Processor: i7-6700K CPU@4.00GHz

No of CPU Cores: 4

Chapter 6 - Experimental Results

The two experiments that are performed on the given dataset are

1) The Million Song Dataset is first passed to threshold blocking Algorithm which performs clustering and assigns cluster numbers. We then calculate mean of all points belonging to same cluster. Above processed data is again passed to K-Means clustering which perform clustering and assigns cluster numbers to the processed data. We now try to identify the final cluster number of each point in the original data based on cluster number assigned for processed data. We call this as hybridized K-Means clustering. Separately the original dataset is passed directly to original K-Means clustering and cluster numbers are assigned directly to the original dataset. We now compare the performance of hybridized K-Means clustering with original K-Means clustering.

2) The Million Song Dataset is first passed to threshold blocking Algorithm which performs clustering and assigns cluster numbers. We then calculate mean of all points belonging to same cluster. Above processed data is again passed to DBSCAN which perform clustering and assigns cluster numbers to the processed data. We now try to identify the final cluster number of each point in the original data based on cluster number assigned for processed data. We call this as hybridized DBSCAN. Separately the original dataset is passed directly to original DBSCAN and cluster numbers are assigned directly to the original dataset. We now compare the performance of hybridized DBSCAN with original DBSCAN.

The threshold blocking algorithm was run for a range of values of k such as 5,10,15,20,50,60,70,80,90,100,150,200,250. Due to computational limits, only some among the k values are chosen to execute in the case of hybridized DBSCAN. This is because, if the centroid data points data set is too large for the clustering algorithm like DBSCAN to run, it takes infinite amount of time to figure out the clusters.

In the case of DBSCAN, a random subset of data is taken to measure the below cluster evaluation measures as well as the performance metrics of the algorithm. While in the case of K-Means, entire Million Song Dataset is considered for analysis.

6.1 Cluster Evaluation Measures

For an unsupervised clustering approach, the evaluation measures are not based on ground truth or on comparison with true label. It is based on the separation of data into clusters. Various indexes and metrics are present to evaluate the performance of the algorithm based on how efficiently an algorithm can separate the data into clusters. Silhouette Coefficient, Calinski-Harabaz Index are some of the examples. Silhouette Coefficient is considered as the standard index among them.

6.1.1 Silhouette Index Value

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). Where, b is the distance between a sample and the nearest cluster that the sample is not a part of. It’s important to know that Silhouette Index and Silhouette Coefficient are synonyms to each other.

The below table summarized the range of values taken by the index value measure when run on the clustering output and the interpretation of value related to the performance of the algorithm.

Range of SC	Interpretation
0.71-1.0	A strong structure has been found
0.51-0.70	A reasonable structure has been found
0.26-0.50	The structure is weak and could be artificial
< 0.25	No substantial structure has been found

6.1.2 Cluster Overlap Measure

Cluster overlap measure determines how many clusters overlap between two clustering algorithm outputs. The overlap of two clusters i.e., cluster1 output from clustering algorithm1 and cluster1 output from clustering algorithm2 is calculated by the number of points in cluster1 of algorithm1 that are also present in the cluster1 of clustering algorithm2.

Given two clustering algorithm outputs, in an unsupervised approach the numbers from both the algorithms do not necessarily talk about the same cluster. For Example, cluster 1 from the output of algorithm1 can relate to the cluster 3 of algorithm 2. In such a case comparing the number of points in Cluster 1 of algorithm1 present in Cluster1 of algorithm2 is not correct and also leads to erroneous results. Also, with each iteration of the algorithm the clusters numbers are randomly assigned to the data set.

To avoid this, the following procedure is carried out in determining the cluster overlap measure: -

1. A matrix is constructed out of clustering algorithm outputs where the row on the top corresponds to cluster number of algorithm1 and column on the left contains the cluster number of algorithm2.
2. The matrix contains values of how many elements match between the cluster outputs from both the algorithms.
3. At the intersection of row *i* and column *j*, the value of the cell *ij* gives the information about how many elements of cluster_i matches with elements of cluster_j.
4. For each row *i* , the maximum value among the intersection of row *i* and various values of column *j* is identified. Cluster *i* and Cluster *j* are assumed to be representing the same cluster.
5. The same process is carried out for rest of the rows as well and the cluster number represented by the row is matched with some column with which it shares maximum number of elements.
6. At the end of nth row, the column value assignments of all the rows have to be distinct. That is each cluster number represented by the column is assigned to one of the cluster number represented by row.

In some cases, at the end of nth row it is possible for one column cluster number to be assigned to more than one row cluster number. It is possible in this case, that a column cluster number is not assigned to any cluster number represented by the row. In such case, use backtracking to assign the column cluster number to row by minimizing the error value. Continue this process until all column cluster numbers are assigned to row cluster numbers and the error is minimized while maximizing the throughput.

Since, the relation between the cluster output labels given by both the algorithms is determined they are compared like in the case of supervised algorithm. One of the cluster outputs is replaced with the mappings obtained from the above algorithm so as to have a baseline to compare both the algorithms. One output of the cluster acts as the ground truth while the other output values of clustering algorithm are evaluated against it. Hence, we obtain the cluster overlap measure between both the clusters.

6.1.3 Cluster Distances

Algorithms that produce clusters with low intra-cluster distances have high intra-cluster similarity and high inter-cluster distances have low inter-cluster similarity. Such a clustering algorithm that produces a collection of clusters having low intra-cluster distance and high inter-cluster distance is considered as the best algorithm based on this criterion.

6.1.3.1 Intra-Cluster Distance

The intra-cluster distance $d'(k)$ is measured as the maximal distance between any pair of elements in cluster k.

6.1.3.2 Inter-Cluster Distance

The inter-cluster distance $d(i,j)$ between two clusters may be any number of distance measures, such as the distance between the centroids of the clusters.

6.2 Experiment 1- Comparing Hybridized K-Means with K-Means

A random subset from 1 Million Song data set is given as input to both Hybridized K-Means algorithm and K-Means Algorithm. The K- value for K-Means denotes the number of clusters to be formed which is 13. Since, the data set contains collection of songs from 13 unique genres as per the source, this value is chosen as K input to K-Means. The output is 13 different clusters where each cluster contains similar songs. The below performance metrics of the algorithm for computation and cluster evaluation are collected over various values of “k”. The “k” value represents the threshold value given to threshold blocking algorithm. When k=0, it implies the data set is run on K-Means itself. The chosen threshold values for the experiment are 15,20,25,50,60,70,80,90,100,150,200,250.

6.2.1 Silhouette Coefficient

Silhouette Coefficient value for K-Means algorithm on the random subset taken is 0.32 which remains same. After K=150, the silhouette value starts dropping. Silhouette Coefficient values above 0.25 is a good indicator to say that cluster output value have high chance of being the actual output value.

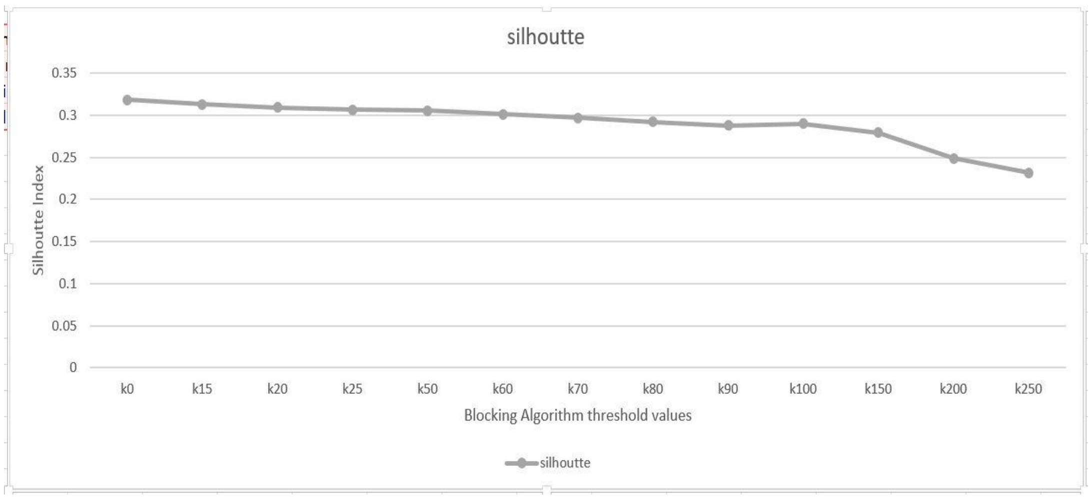

Figure 2 Silhouette Coefficient for K-Means on data subset

6.2.2 Intra-Cluster Distance

Average intra cluster distance is taken and plotted against various values of k. For k=0 i.e., on original K-Means algorithm, the value is around 70. This average distance increases initially until threshold value k =15 which is being run on hybridized K-Means algorithm, but later decreases and remains the same having a value around 40. This indicates there is good high intra-cluster similarity in the clusters obtained from the hybridized algorithm when compared to original K-Means algorithm.

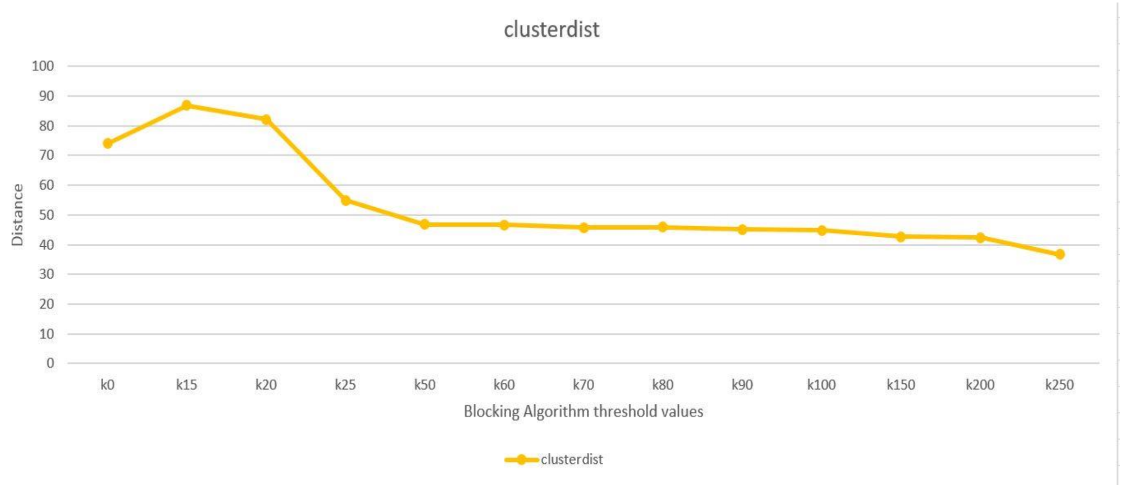

Figure 3 Intra Cluster distance for K-Means on data subset

6.2.3 Inter-Cluster Distance

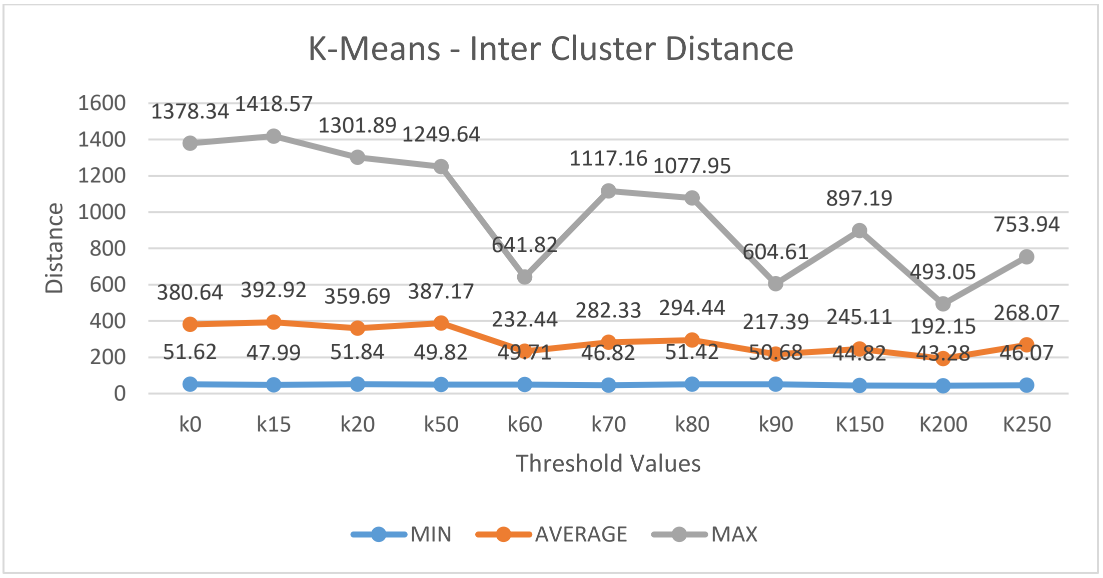

Figure 4 Inter-Cluster distance for K-Means on data subset

6.2.4 Cluster Overlap

On an average, 80% of the clusters formed by threshold blocking algorithm are overlapping with K-Means algorithm.

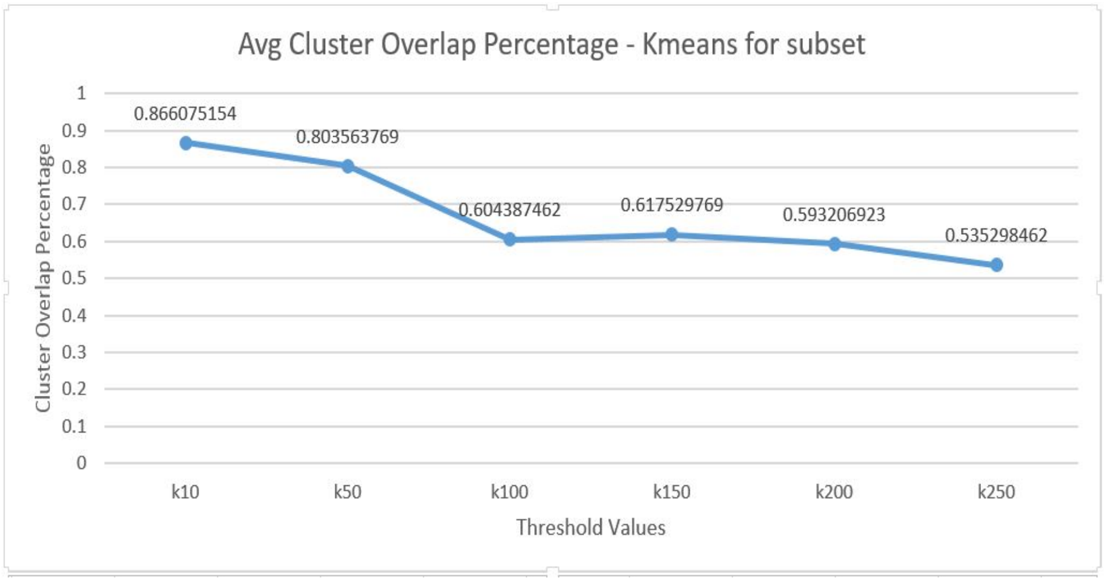

Figure 5 Cluster Overlap percentage for K-Means on data subset

6.2.5 Processing Time

The processing time of the hybridized K-Means algorithm remains low when compared with K-Means algorithm until a certain threshold value k=90. But there is only a significant drop in the processing time below k=50. This could be because, with increase in k value the time taken to form blocks with nearest neighbors increases.

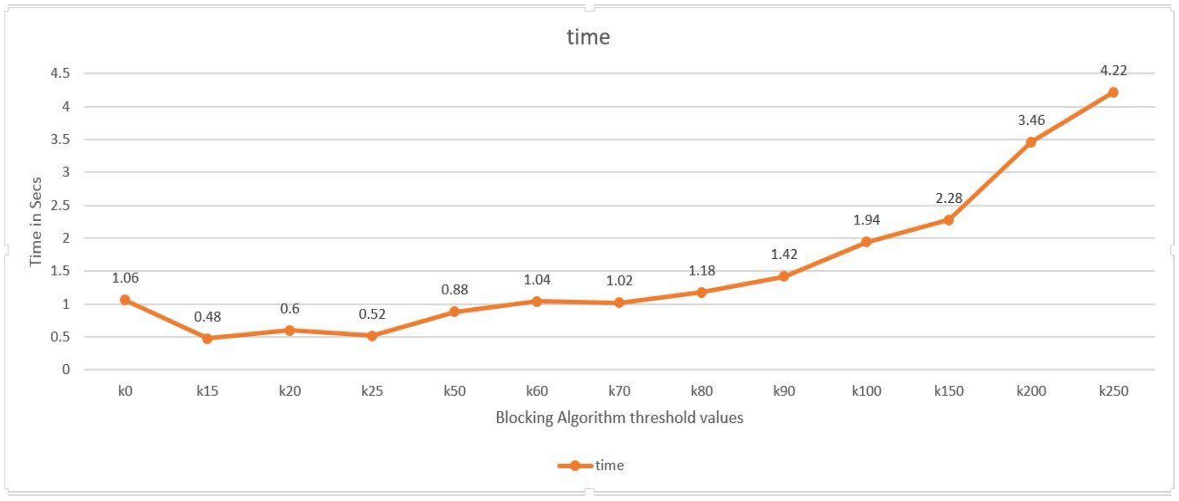

Figure 6 Processing Time for K-Means on data subset

6.2.6 Memory

The memory required to run the hybridized K-Means algorithm is comparatively low when compared to the memory required to run the Original K-Means algorithm on the dataset. This could be because, the size of dataset reduces after the formation of blocks by threshold blocking algorithm and only the centroids are given to K-Means after that step. The threshold blocking algorithm do not seem to occupy much memory to form the blocks.

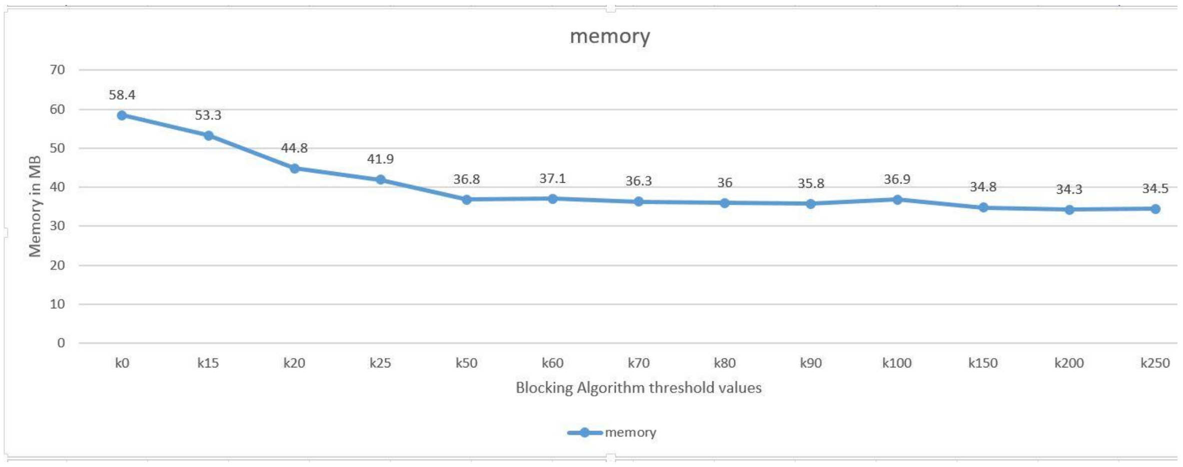

Figure 7 Memory for K-Means on data subset

The same experiment is carried out on 1Million Dataset. Due to computational limits, intra-cluster, inter-cluster and silhouette measures are not calculated but memory, processing time and cluster overlap are calculated.

6.2.7 Cluster Overlap for K-Means on 1M dataset :-

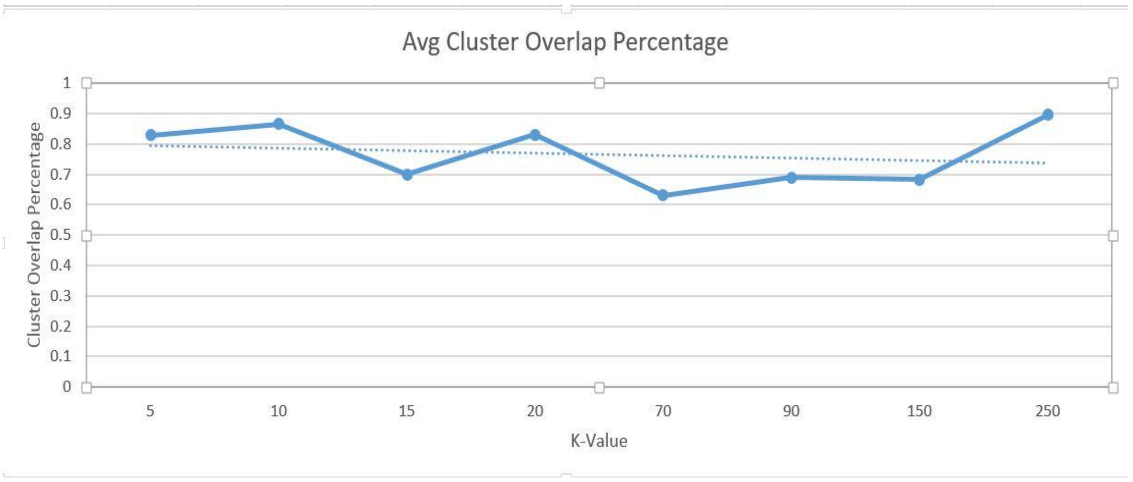

Figure 8 Cluster Overlap for K-Means on 1M data

6.2.8 Processing Time

There is an increasing in the processing time slightly until a certain value of k, but later has a sharp increase in the processing time. This could be because it might be taking longer time to find similar samples in the dataset for large values of k.

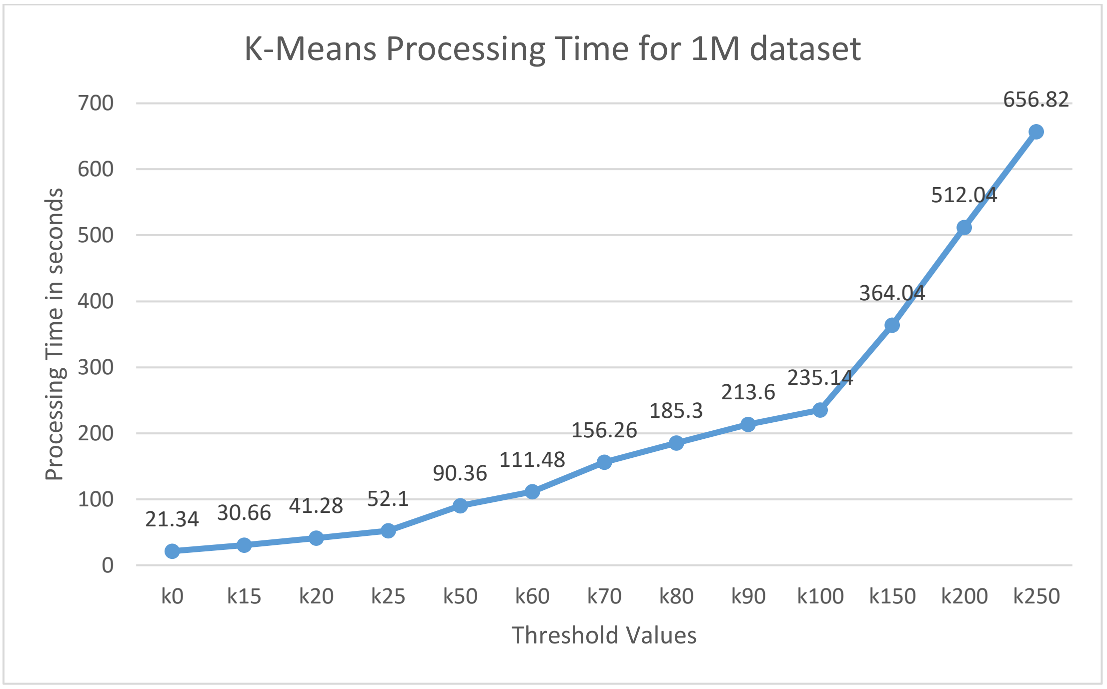

Figure 9 Processing Time for K-Means on 1M data

6.2.9 Memory

There is drop in the memory required for computation with increasing value of K since k-means has fewer points to cluster upon with increasing values of k.

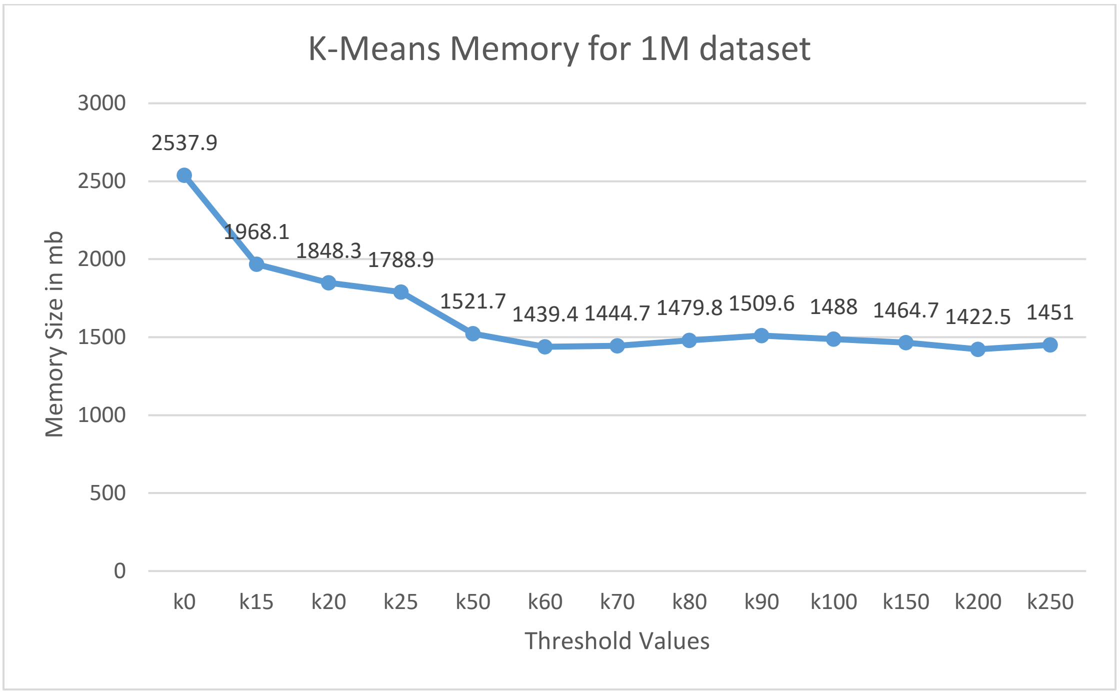

Figure 10 Memory required for K-Means on 1M data

6.3 Experiment 2 - Comparing Hybridized DBSCAN with DBSCAN

A random subset from 1 Million Song data set is given as input to both Hybridized DBSCAN algorithm and DBSCAN Algorithm. The value of eps and minPts is determined for each experiment. This is because with varying threshold value given to threshold blocking algorithm, the number of data points that will be given to DBSCAN varies. So, for each value of threshold value “K” knndistplot to estimate the parameters is drawn. MinPts is equal to “k” value given in knndistplot while eps value is obtained from the bend of the curve in the plot. The output of the algorithm is 13 different clusters where each cluster contains similar songs. The below performance metrics of the algorithm for computation and cluster evaluation are collected over various values of “k”. The “k” value represents the threshold value given to threshold blocking algorithm. When k=0, it implies the data set is run on DBSCAN itself. The chosen Threshold values for the experiment are 10,20,25,50,60.

6.3.1 Silhouette Coefficient

Silhouette Coefficient value for DBSCAN algorithm on the random subset taken is 0.9. Silhouette coefficient values do not seem to follow a pattern but on an average the value stays between 0.5 to 0.6. Though there is a drop in value, the value still lies above 0.5 ensuring that the cluster output is an acceptable result.

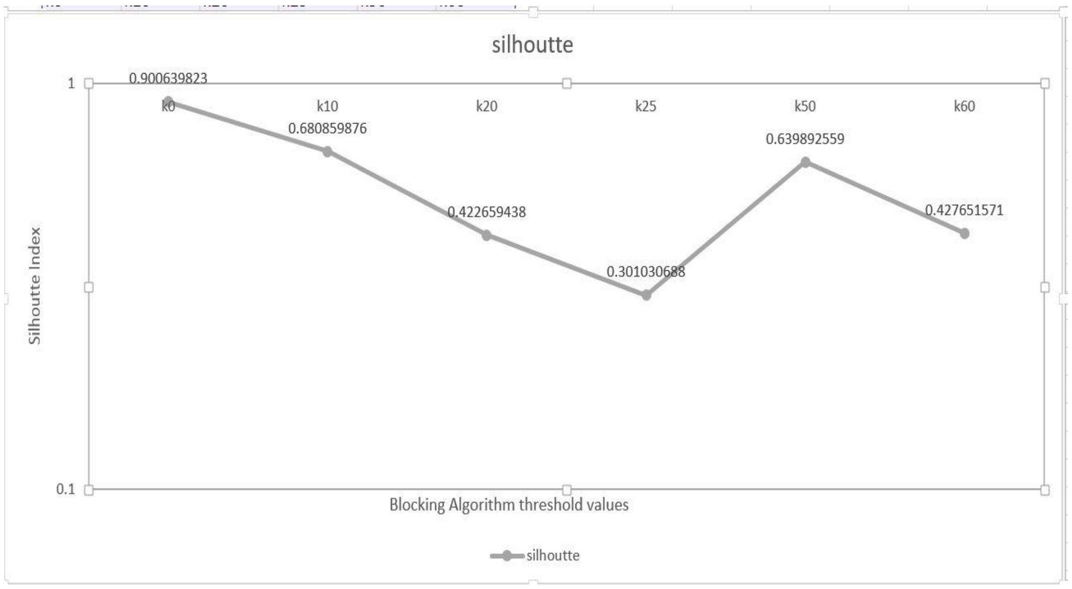

Figure 11 Silhouette Coefficient value for DBSCAN on data subset

6.3.2 Intra-Cluster Distance

Average intra cluster distance is taken and plotted against various values of k. For k=0 i.e., on original DBSCAN algorithm, the value is around 360 . This average distance decreases and remains at an average value of 200 with various values of K. This indicates there is good high intra-cluster similarity in the clusters obtained from the hybridized algorithm when compared to original DBSCAN algorithm.

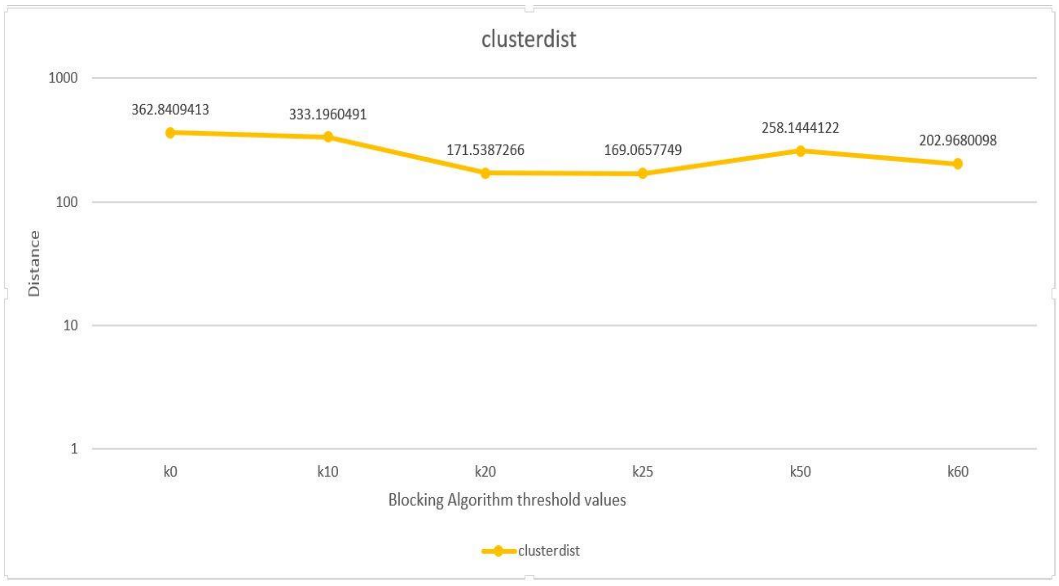

Figure 12 Intra Cluster distance for DBSCAN on data subset

6.3.3 Inter-Cluster Distance

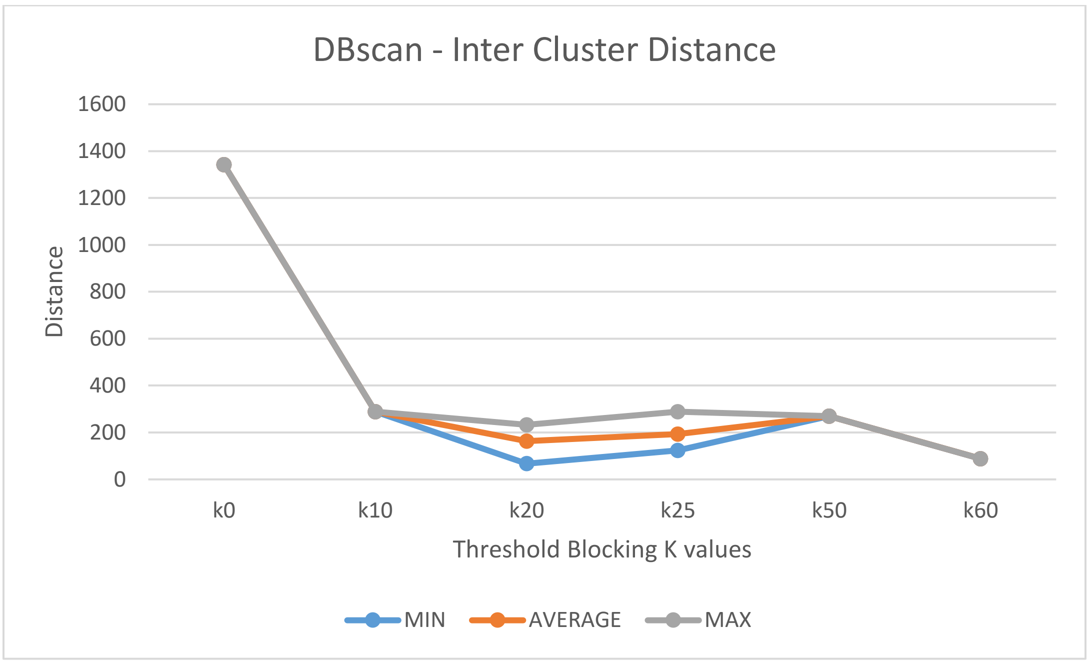

Figure 13 Inter Cluster distance for DBSCAN on data subset

6.3.4 Cluster Overlap

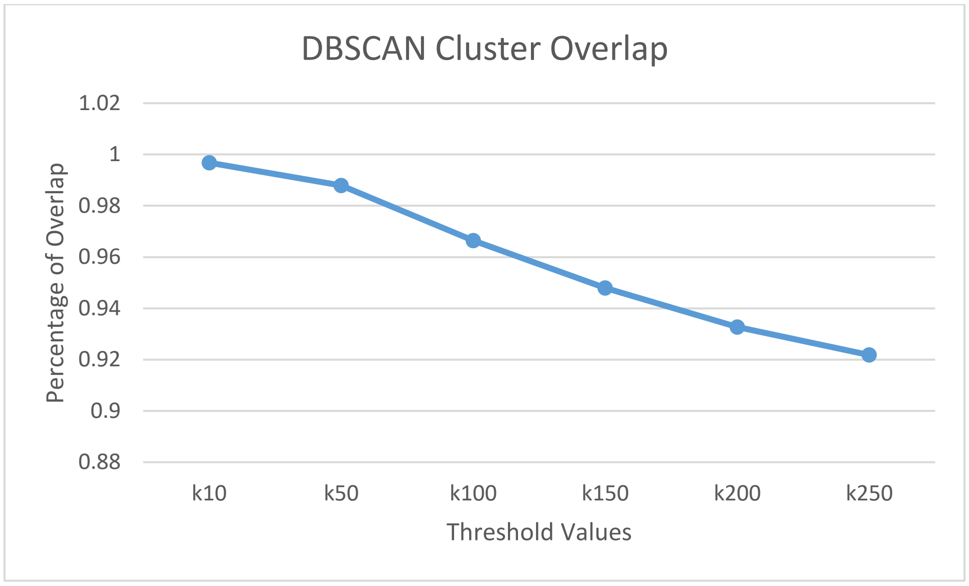

Figure 14 Cluster Overlap measure for DBSCAN on data subset

6.3.5 Processing Time

The processing time of the hybridized DBSCAN algorithm remains significantly low when compared with DBSCAN for all values of K. Also, this increases the DBSCAN capability of clustering large data sets.

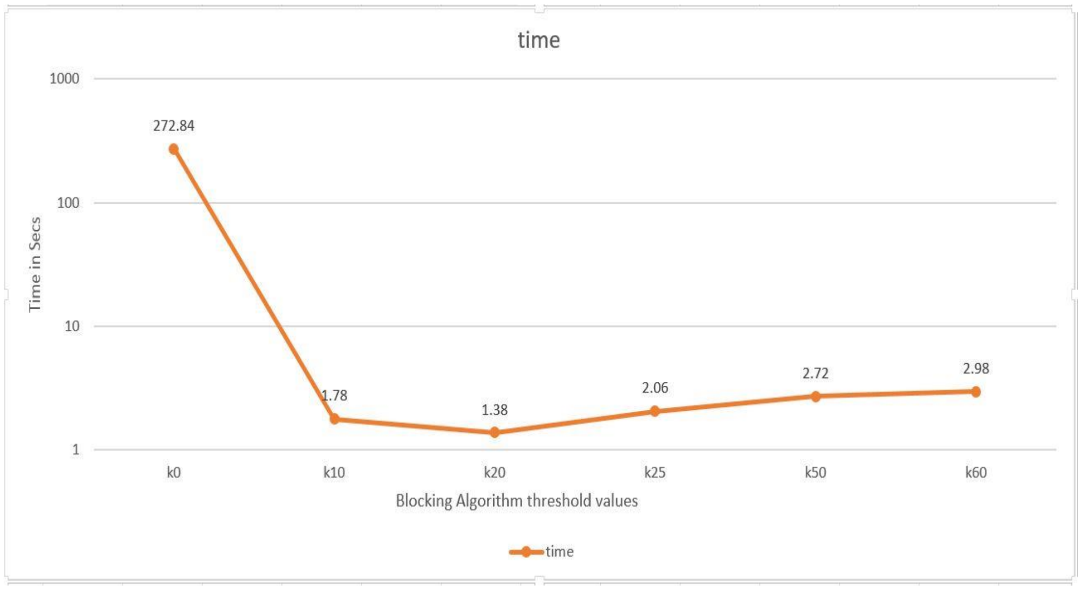

Figure 15 Processing Time for DBSCAN on data subset

6.3.6 Memory

The memory required to run the hybridized DBSCAN algorithm is comparatively low when compared to the memory required to run the Original DBSCAN algorithm on the dataset. This could be because, the size of dataset reduces after the formation of blocks by threshold blocking algorithm and only the centroids are given to DBSCAN after that step. The threshold blocking algorithm do not seem to occupy much memory to form the blocks.

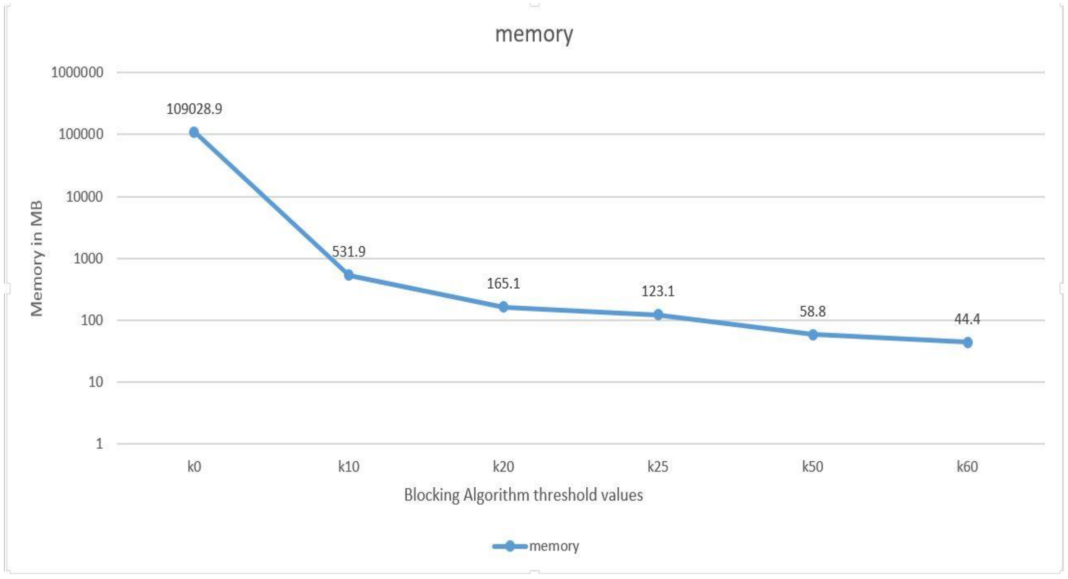

Figure 16 Memory for DBSCAN on data subset

Chapter 7 - Summary and Future Work

7.1 Summary

Hybridized K-Means and DBSCAN algorithm are proved to perform better when compared with Original DBSCAN and K-Means algorithm. In terms of memory or processing time, the hybridized algorithms show a significant drop which indicates the capability of threshold blocking to be extensible to perform (Higgins, 2016) clustering on large datasets. DBSCAN is initially not feasible to execute on Million song dataset while also taking large amount of time to run on the subsets of the data. When combined with threshold blocking algorithm, there is drop in the time and memory taken for execution without affecting the efficiency of the clustering Algorithm. The same applies to K-Means and Hierarchical Agglomerative Clustering. This is tested for 1 million data but the experiments are limited by computing demand of algorithms which calculates metrics on the output of clusters. The distance metric that is calculated for 1 million data requires a huge RAM around 2500 GB. Ideally, such an amount of computing memory is not required to generate clusters or for calculating metrics on 1 million data set. To avoid this problem, the computation of metrics, cross-validation have to be calculated using map-reduce algorithms executed on big data technologies like Hadoop, Spark and soon. The performance of threshold blocking algorithm in association with clustering algorithms is tested over various values of K but the performance of threshold blocking algorithm for different sizes of data sets is yet to be explored. The limit of dataset size that threshold blocking algorithm can efficiently handle needs to be calculated.

7.2 Future Work

Even though the experiment helped us to understand better about the efficiency of hybridized algorithm on large dataset, the research can be expanded in many other ways. Map Reduce Framework is helpful to overcome the memory issues that will arise while calculating the clustering evaluation metrics for large datasets. In addition, threshold blocking algorithm can be executed iteratively using large and small values of k to measure the variation in performance. Further, it would be interesting to see how threshold blocking algorithm works on various data sizes.

References

Abbas, O. A. (2008). Comparisions Between Data Clustering Algorithms. *The International Arab Journal of Information Technology*. Retrieved April 10, 2017

Annakula, C. (2017). *Hierarchial and Partitioning-Based Hybridized Blocking Model.* M.S.Report, Kansas State University. Retrieved April 10, 2017

Cavan, R., Changchun, W., & Mark, R. (2005). A RAPID METHOD FOR THE COMPARISON OF CLUSTER ANALYSES. *Statistica Sinica*, 19-33. Retrieved April 10, 2017

Cluster Analysis. (2017). Retrieved April 10, 2017, from Wikipedia: https://en.wikipedia.org/wiki/Cluster_analysis

Cluster Performance Evaluation. (2017). Retrieved April 10, 2017, from Scikit Learn: http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

Cross Validation. (2017). Retrieved April 10, 2017, from Wikipedia: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Determining the number of clusters in a data set. (2017). Retrieved April 10, 2017, from Wikipedia: https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

Enrique, A., Julio, G., Javier, A., & Felisa, V. (2009). A comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints. Retrieved April 10, 2017

Higgins, M. J., Savje, F., & Sekhon, J. S. (2016). Improving massive experiments with threshold. *PNAS*. Retrieved April 10, 2017

K-Means Clustering. (2017). Retrieved April 10, 2017, from Wikipedia: https://en.wikipedia.org/wiki/K-means_clustering

Lamere, T. B.-M. (2011). *Million Song Dataset*. Retrieved April 10, 2017, from The Million Song Dataset: https://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset

Martin, E., Hans-Peter, K., Jiirg, S., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters. *AAAI*. Retrieved April 10, 2017

R.Lleti, M.C.Ortiz, Sarabia, L., & Sanchez, M. (2004). Selecting variables for K-Means cluster analysis by using a genetic algorithm that optimises the silhouettes. *Analytica Chimica*, 87-100. Retrieved April 10, 2017

Rousseeuw, P. J. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 53-65. Retrieved April 10, 2017

Silhouette Clustering. (2017). Retrieved April 10, 2017, from Wikipedia: https://en.wikipedia.org/wiki/Silhouette_(clustering)

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining.* Retrieved April 10, 2017

Appendix A - Attributes of the Dataset

artist_mbid: db92a151-1ac2-438b-bc43-b82e149ddd50

the musicbrainz.org ID for this artists is db9...

artist_mbtags: shape = (4,)

this artist received 4 tags on musicbrainz.org

artist_mbtags_count: shape = (4,)

raw tag count of the 4 tags this artist received on musicbrainz.org

artist_name: Rick Astley

artist name

artist_playmeid: 1338

the ID of that artist on the service playme.com

artist_terms: shape = (12,)

this artist has 12 terms (tags) from The Echo Nest

artist_terms_freq: shape = (12,)

frequency of the 12 terms from The Echo Nest (number between 0 and 1)

artist_terms_weight: shape = (12,)
weight of the 12 terms from The Echo Nest (number between 0 and 1)
audio_md5: bf53f8113508a466cd2d3fda18b06368
hash code of the audio used for the analysis by The Echo Nest
bars_confidence: shape = (99,)
confidence value (between 0 and 1) associated with each bar by The Echo Nest
bars_start: shape = (99,)
start time of each bar according to The Echo Nest, this song has 99 bars
beats_confidence: shape = (397,)
confidence value (between 0 and 1) associated with each beat by The Echo Nest
beats_start: shape = (397,)
start time of each beat according to The Echo Nest, this song has 397 beats
danceability: 0.0
danceability measure of this song according to The Echo Nest (between 0 and 1, 0 => not analyzed)
duration: 211.69587

duration of the track in seconds
end_of_fade_in: 0.139
time of the end of the fade in, at the beginning of the song, according to The Echo Nest
energy: 0.0
energy measure (not in the signal processing sense) according to The Echo Nest (between 0 and 1, 0 => not analyzed)
key: 1
estimation of the key the song is in by The Echo Nest
key_confidence: 0.324
confidence of the key estimation
loudness: -7.75
general loudness of the track
mode: 1
estimation of the mode the song is in by The Echo Nest
mode_confidence: 0.434
confidence of the mode estimation

release: Big Tunes - Back 2 The 80s
album name from which the track was taken, some songs / tracks can come from many albums, we give only one
release_7digitalid: 786795
the ID of the release (album) on the service 7digital.com
sections_confidence: shape = (10,)
confidence value (between 0 and 1) associated with each section by The Echo Nest
sections_start: shape = (10,)
start time of each section according to The Echo Nest, this song has 10 sections
segments_confidence: shape = (935,)
confidence value (between 0 and 1) associated with each segment by The Echo Nest
segments_loudness_max: shape = (935,)
max loudness during each segment
segments_loudness_max_time: shape = (935,)
time of the max loudness during each segment
segments_loudness_start: shape = (935,)

loudness at the beginning of each segment

segments_pitches: shape = (935, 12)

chroma features for each segment (normalized so max is 1.)

segments_start: shape = (935,)

start time of each segment (~ musical event, or onset) according to The Echo Nest, this song has 935 segments

segments_timbre: shape = (935, 12)

MFCC-like features for each segment

similar_artists: shape = (100,)

a list of 100 artists (their Echo Nest ID) similar to Rick Astley according to The Echo Nest

song_hotttnesss: 0.864248830588

according to The Echo Nest, when downloaded (in December 2010), this song had a 'hotttnesss' of 0.8 (on a scale of 0 and 1)

song_id: SOCWJDB12A58A776AF

The Echo Nest song ID, note that a song can be associated with many tracks (with very slight audio differences)

start_of_fade_out: 198.536

start time of the fade out, in seconds, at the end of the song, according to The Echo Nest
tatums_confidence: shape = (794,)
confidence value (between 0 and 1) associated with each tatum by The Echo Nest
tatums_start: shape = (794,)
start time of each tatum according to The Echo Nest, this song has 794 tatums
tempo: 113.359
tempo in BPM according to The Echo Nest
time_signature: 4
time signature of the song according to The Echo Nest, i.e. usual number of beats per bar
time_signature_confidence: 0.634
confidence of the time signature estimation
title: Never Gonna Give You Up
song title
track_7digitalid: 8707738
the ID of this song on the service 7digital.com
track_id: TRAXLZU12903D05F94

The Echo Nest ID of this particular track on which the analysis was done
year: 1987
year when this song was released, according to musicbrainz.org

Since the project aims to identify similar songs to group them into genres, only few fields among all the above fields are sufficient for the task. Loudness, Tempo, Time_Signature, Duration and Key are the fields that will be used in this project. So, the million records consisting only these fields is used in the experiment.