DISTRIBUTED AND PROGRESSIVE FEATURE SELECTION ALGORITHM FOR HIGH DIMENSIONAL DATA: A MAP-REDUCE APPROACH

[1]CH. RAJA RAMESH, [2]G. JENA[3]K RAGHAVA RAO
[1]Research Scholar, K L University, Associate Professor, Sri Vasavi Engineering College, Tadepalligudem,chrajaramesh@gmail.com, [2]professor & HOD, Dept. of CSE, BVC Engg. College, [3]Professor,K L University,

ABSTRACT

Dimensionality reduction or feature selection is an essential pre-processing step to apply machine learning algorithm further on any data set. But at for medium dimensional datasets it is optional or on-demand requirement. But it is mandatory in high dimensional datasets. Its significance is increased to get the accurate and relevant output from machine learning algorithm. Most of the existing methods are divided into 2 types one is Dimensionality reduction and the other one is feature selection. There is very narrow gap between these two methods. Dimensionality reduction is more mathematical analysis with transformations and may or may not have same subset of features from original features. Feature selection is application of feature engineering and requires domain knowledge. But any algorithm applicable for high dimensional data requires more processing time and storage resources. We considered the processing time as basis for our problem statement and implemented a distributed algorithm for Feature Selection and named as Distributed Progressive Feature selection algorithm with Knn+Relieff for high dimensional data. In this paper applied MapReduce concept to select final sub set of relevant features in progressive manner. Simulation results showthe feature with its weights for various parameters.

Keywords: *Feature Selection, Dimensionality Reduction, Mappers, Similarity Measures*

1. INTRODUCTION

To implement machine learning and statistics **dimensionality reduction** or reduce the dimension is the process of reducing the number of unwanted variables under consideration,[1] via obtaining a set of principal variables. It can be divided into feature selection and feature extraction in machine learning and statistics, dimensionality reduction or dimension reduction is the method of reducing the amount of random variables into consideration [1] via getting a collection of principal variables. It is often divided into feature choice and have extraction

2.1 Feature Selection

Approaches try to find a subset of the original variables (also called features or attributes). There are three strategies; *filter* (e.g. information gain) and *wrapper* (e.g. search guided by accuracy) approaches, and *embedded* (features are selected to add or be removed while building the model based on the prediction errors). See the problems, in some cases, data analysis such as regression or classification can be done in the reduced space more accurately than in the original space.

In machine learning and measurements, include determination, otherwise called variable choice, quality choice or variable subset choice, is the way toward choosing a subset of applicable highlights (factors, indicators) for use in display development. Highlight choice procedures are utilized for four reasons:

- Improvement of models to make them simpler to translate by analysts/users
- Shorter preparing times,
- To avoid the scourge of dimensionality,
- Improved speculation by dimentioning over fitting

2.2 Feature Extraction

Feature extraction is reducing the sufficient amount of resources required to describe a huge datasets. While performing analysis process the difficult data one of the main problem stems from the number of elements involved. To process or analyze the number of variables it requires large amount of resources (memory, computational capacity of system) are required. Highlight extraction is a general term for strategies for building blends of the factors to get around these issues while still depicting the information with sufficient accuracy. Results can be enhanced utilizing developed arrangements of use subordinate highlights, ordinarily worked by a specialist. One such process is called include building. Sometimes use the dimensionality reduction techniques also. Some of the techniques are

1. Independent Component analysis
2. Kernel PCA
3. Latent Symantec Analysis
4. Principal component analysis
5. Partial least squares
6. Nonlinear dimensionality reduction etc.

Transform the data from high dimensional space to a space of fewer dimensions. The data transformation may be linear, as in principal component analysis (PCA), but many nonlinear dimensionality reduction techniques also exist.[3][4] For multidimensional data, tensor representation can be used in dimensionality reduction through multilinker subspace learning.

2.3 Feature Support Count

Feature Count allows you to view the number of features in the map based on feature classes and subtypes, respectively. A total number is given for each feature class, then for each subtype. The grand total of all the features in the map is displayed at the bottom of the window. The total feature count provides a snapshot of the features that are currently loaded in the map. The way the feature classes are listed in the Total Feature Count window matches the table of contents, and each feature class can be expanded or collapsed to view quantities for individual subtypes.

2. RELATED WORK

3.1 Theoretically Optimal Feature Selection

The "optimal feature selection" framework [2], initially, places a sound theoretical foundation for the selecting features are the main task. Based on the existing data theory, this framework defines the optimality of a feature set within the sense that it retains the foremost quantity of data needed for modeling the dependence between the input variables (features) and output variable (label) within the reduced-dimensional space. Let T(x) denote the illustration of x when the spatial property reduction outlined by T this framework needs that the posterior p(y|T(x)) be as shut as attainable to the first one p(y|x)

3.2 Feature Weighting Relief [3]

The computational issue of combinational search can be some extent to be alleviating by using a feature weighting strategy. By using feature weights consider as real-valued numbers instead of binary ones enables the employment of some well-established optimization techniques and, thus, it allows for implementation of efficient algorithmic. Among the existing feature weighting algorithms, the RELIEF algorithm [4] is considered one of the most successful ones due to its simplicity and effectiveness [5]. Algorithm pseudo code is presented on reference [4]. The key idea of RELIEF is to iteratively estimate feature weights according to their ability to discriminate between neighboring patterns. In each iteration, a pattern x is randomly selected and then two nearest neighbors of x are found, one from the same class

3.3 Contribution of Mr. Yijun Sun

First, in algorithmic aspects, starting from a new interpretation of RELIEF, we propose a set of feature weighting algorithms. The effectiveness of those algorithms, in terms of solution quality and computational efficiency, is experimentally demonstrated on a wide variety of data sets. Considering the increased demand for analyzing data with large feature dimensionality in some emerging domains such as bioinformatics, we expect widespread usage of these algorithms in these applications.

Second, in theoretical aspects, this paper may provide a new direction of feature selection research in addition to providing some new algorithms. Feature selection plays a critical role in machine learning. Yet, as opposed to classifier design, it still to date lacks rigorous theoretical treatment. This is largely due to the difficulty in defining an objective function that can be easily optimized by some well established optimization techniques. It is particularly true for wrapper methods that use a nonlinear classifier to evaluate the goodness of selected feature subsets. The crisp partition of a feature set and the nonlinearity of a classification function make the resulting objective function non convex and even non differentiable. For this reason, most feature selection algorithms rely on heuristic search. The I-RELIEF algorithms has a clearly defined objective function and can be solved through numerical analysis instead of combinatorial search and, thus, presents a promising direction for a more rigorous treatment of feature selection problems.

4. EXISTING KNN RELIEF ALGORITHM

4.1 Introduction To KNN

Knn is a non parametric lazy learning algorithm that could be a quite concise declaration. When you say a way is non parametric, it way that it does not make any assumptions on the underlying information distribution [6]. This is quite useful, as within the real global; maximum of the practical statistics does not obey the everyday theoretical assumptions made (eg. Gaussian mixtures, linearly separable and so on). Non parametric algorithms like KNN come to the rescue here.

It is also a lazy algorithm. What this means is that it does now not use the schooling records points to do any generalization. In other phrases, there's no express education segment or its miles very minimal. This means the education section is quite rapid. Lack of generalization means that KNN keeps all of the education information more precisely, all of the schooling facts is wanted throughout the checking out phase. (Properly this is an exaggeration, however now not far from fact). This is in contrast to other strategies like SVM where you may discard all non assist vectors with none problem. Maximum of the lazy algorithms particularly KNN makes selection based on the whole schooling data set (inside the first-rate case a subset of them).

The dichotomy is pretty obvious right here there is a nonexistent or minimum training section but a high priced checking out section. The fee is in terms of each time and reminiscence. More time is probably needed as in the worst case, all information points might take point in selection more reminiscence is needed as we want to keep all schooling records.

When we are using KNN it contains some assumptions also KNN assumes that the data is a feature space it exactly the data points are ina metric space. Most of the times data can be scalars or possibly multidimensional space, even the points are in feature space, they have a notion of distance, This need no longer always be Euclidean distance although it is the one typically used. Every set of the training data associated with each vector. In the easiest case it will be either positive classes or negative classes [6].

4.2. Introduction to Relief Algorithm

Relief [8] (algorithm is originally proposed by Rendell & Kira) is an characteristic reweighting algorithm employed successfully in various diverse setting. It learn a vector of weights for each of the different attributes or features describing their importance. It has been proved by Wu and Sun that it absolutely target at maximizing the margin of liner utility function.

SiLa is a similarity metric learning algorithm for nearest neighbor classification. It aim at moving the nearest neighbors belonging to the same class nearest to the given input example (termed as target neighbors) while inserting away the nearest examples belong to different classes (describe as impostors). The similarity function between two examples x and y can be written as

$$S_A(x,y) = \frac{x^t A_y}{N(x,y)}$$

Here t represents transpose, A is (p X P) similarity of matrix and N(x,y) is a normalization function which depends on y and x (this normalizations is typically used to map the similarity function to a particular interval, a [0,1]. The above equation generalizes several standard similarity functions e.g. the cosine measure which is widely used in text retrieval, is obtained by setting the matrix to the identity matrix and N(x,y) to the product of the L2 norms of x and y. The aim here is to reduce the 0 -1 loss which is dependent on the number of mistakes made during the classification phase. For the remainder of the paper, a matrix is few times represented as a vector as well (example a p X P matrix can be represented by a vector having p^2 elements).

4.3. RELIEF and its mathematical interpretation

Sun and Wu [12] have shown that the RELIEF algorithm solves convex optimization problem while maximizing a margin-based objective function employing the Knn algorithm. It learns a vector of weights for each of the features, based on the nearest hit (nearest example belonging to the class under consideration, also known as the nearest target neighbor) and the nearest miss (nearest example belonging to other classes, also known the nearest impostor). In the original setting for the RELIEF algorithm, it only learns a diagonal matrix. However, Sun and Wu [12] have learned a full distance metric matrix and have also proved that RELIEF is basically an online algorithm. In order to describe the RELIEF algorithm, we suppose that x (i) is a vector in Rp with y (i) as the corresponding class label with values +1, −1. Furthermore, let A be a vector of weights initialized with 0. The weight vector learns the qualities of the various attributes. A is learned on a set of training examples. Suppose an example x (i) is randomly selected. Then two nearest neighbors of x (i) are found: one from the same class (termed as the nearest hit or H) while the second one from a class other than that of x (i) (termed as the nearest miss or M). The update rule in case of RELIEF does not depend on any condition unlike SiLA.

4.4 Strength and Limitation of Existing KNN

There is a theoretical guarantee that with a huge dataset and large values of k, you're going to get good results from nearest neighbor learning. Nearest neighborhood methods can be lousy when p (the number of variable) is large because of the curse of dimensionality. In high dimension, it's really difficult to stay local.

The main limitation of the Knn is to make each prediction scan the entire training data set is very slow. To avoid this program we are going to implement MapReduce method by using Knn relief.

5. MAPREDUCING

MapReduce is the heart of Hadoop. It's far this programming paradigm that permits for big scalability across masses or lots of servers in a Hadoop cluster. The MapReduce idea is reasonably easy to recognize for those who are acquainted with clustered scale-out records processing answers.

For human beings new to this subject matter, it can be incredibly tough to grasp, as it's now not generally something humans have been exposed to previously. If you're new to HadoopMap Reduce jobs, don't worry: we're going to describe it in a manner that receives you on top of things quickly.

The term MapReduce in reality refers to 2 separate and distinct duties that Hadoop packages perform. The first is the map job, which takes a fixed of statistics and converts it into another set of statistics, in which person elements are broken down into tuples (key/cost pairs). The lessen activity takes the output from a map as enter and combines the ones information tuples right into a smaller set of tuples as the collection of the name Map Reduce. It implies, the reduce activity is usually achieved after the map job.
Generally MapReduce process is based on pass to the computer where the data resides. Program execute in three stages. 1. Map stage 2. Shuffle stage 3. Reduce stage.

5.1 Map Stage

In this stage mappers main job is to process the input data. This data is in the form of file or directory and stored in the HDFS. The input data file or data directory is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data

5.2 Reduce Stage

In reduce stage combination of the **Shuffle** stage and the **Reduce** stage. The main job of the reducer is to process the data that come from mapper. After completion of this stage it produces the new set of data output and send to the HDFS.

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

- This Hadoop framework manages all the details of data-passing such as issuing jobs, verifying job completion, and copying data around the cluster between the different nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given job, the cluster collects and reduces the data to form an appropriate result, and sends it back to the server.

6. PROPOSED KNN RELIEF ALGORITHM IMPLEMENTATION USING MAPREDUCERS

No single method gives accurate results or it is not good practice to depend on single methods result. Because it may or may not fit to all type of data. Computing and space complexity also come in account when deal with large data sets and data streams. Thus in any aspect selection of more than one method and aggregate the results or use the majority voting of those methods.

Thus our proposed system uses ensemble approach. It is also having some more capabilities to handle with large and very high dimensional data sets. Those are, make the algorithm as parallel, distributed and evolutionary. Parallelism is achieved through concurrent programming to fully utilize the CPU with the support of core processors. Distributed nature is achieved through MapReduce based implementation and finally Genetic algorithm is used as evolutionary computing method to automate the selection process without manual intervention in parameter tuning and cluster analysis.

More over results are aggregated from all of the approaches by selecting the intersection of features generated from various methods. These features in turn applied to fuzzy clustering algorithm and evaluate the cluster quality. This process is repeated till final set of relevant features are selected. This is a onetime process. Once final set of features are selected and all the other features are eliminated computation, reprocessing and space complexity will be reduced and also any clustering algorithm not only fuzzy clustering gives good results.

We use two types of dimensionality reduction techniques. One is non-linear based kernel functions and other is purely statistical approach. Technically these two techniques are fully diversified methods. Thus more relevant features which are fit in all aspects are only selected with proposed approach.

1. Algorithm Steps

```
0.Partition high dimensional input file into multiple files using vertical partition and place them in input folder.
1.Setup project properties such as thresholds, input, output folders
2.Build Map-Reduce Environment using RMI
        2.1 Build One Mapper for each input file
        2.1.1 Build Dataset at each mapper from respective mappers input file
        2.1.2 Find Min and Max of each feature
        2.1.3 Find normalized dataset
        2.1.4 Execute Knn+Relieff Algorithm for Feature Selection
        2.1.5 Build Dataset with selected features
        2.2 Build Reducer by reading all the datasets generated from all mappers
        2.2.1 Build the dataset at reducer with union of features and data instances extracted
from above reduced data sets from mappers
        2.2.2 Again apply kNN+ReliefF Algorithm on this dataset
        2.2.3 Build Reduced Dataset with selected features
3.Now Save the selected features in a list
4. Repeat Steps 2.1,2.2 and 3 for given number of times (from properties file) (Evolutionary
computing step)
5. Find the frequency of each feature after all iterations
6. Find the support of each feature or dimension
7. Mark the features with support > given threshold (from properties file)
8. Build the dataset from finally marked features
9. Apply Clustering on dataset with reduced features
10. Find the quality of clusters
11. Build the dataset with all the features/dimensions
12. Apply Clustering on whole dataset
13. Find the cluster quality
14. Compare the quality of these two methods.
15. Find the time and space complexity for both sequential and distributed map-reduce frame
work for comparison
```

7. SYSTEM ENVIRONMENT AND RESULTS DISCUSSION

System Environment used for execution of above algorithm includes Windows 8.1 64-bit OS with JDK 7. Implementation of algorithm with RMI and resembles Map-Reduce in Hadoop. But it is more light weight than and designed specific to domain what current work focuses on. These environments easily setup on any system with normal configuration with open source software. There are following points are involved in results.

7.1 Clustering Without Feature Selection

All the dimensions or features influence grouping of instances in given data set. But inclusion of all the features to clustering process means, you are trying to find the best partner with full and exact similarity with all your features. It reduces or sometimes don’t give the scope of grouping or clustering. In that case number clusters will be formed and most of the time it will be cluster for namesake but internally will contain only single instance. Then we can say all the instances are outliers. Then there is no need to apply clustering as well. This is one face of a coin. Other thing need to be considered is, how much extent each feature’s impact in clustering. To understand that we ran the clustering algorithm without feature selection. After that removed some of the features manually and observed the output. We found significant changes in output of clustering. Removing of one set of the features gave more impact and some gave less impact on the clustering. This practice as basis for auto feature selection in high dimensionality data set. Inclusion of all the features increases runtime complexity and decreases quality of the cluster as well.

7.2 Clustering with Automated Feature Selection

As mentioned in above paragraph, identifying and removing of features manually in high dimensional data set is most difficult and time consuming and error prone. Similarly executing feature selection algorithm for high dimensional dataset in a single system is also time consuming. To address these issues, convert feature selection algorithm as parallel algorithm with map-reduce framework. Here also we considered the bad experiences found by number of researchers in the literature regarding the accuracy of single algorithm. Kept this point in mind, we implemented ensemble features selection methods with parallel execution.

7.3 Without Feature Support Count

We calculated weight for each feature based on support count using evolutionary computing by varying thresholds and cluster quality as objective function. But no features are pruned from final set of features after feature selection to understand basic feature selection accuracy.

7.4 With Feature support count

In this step some of the features are pruned based on support count from the output of evolutionary features selection process and input the final feature set (after pruning) to cluster and compared the results in terms of cluster quality and time complexity.

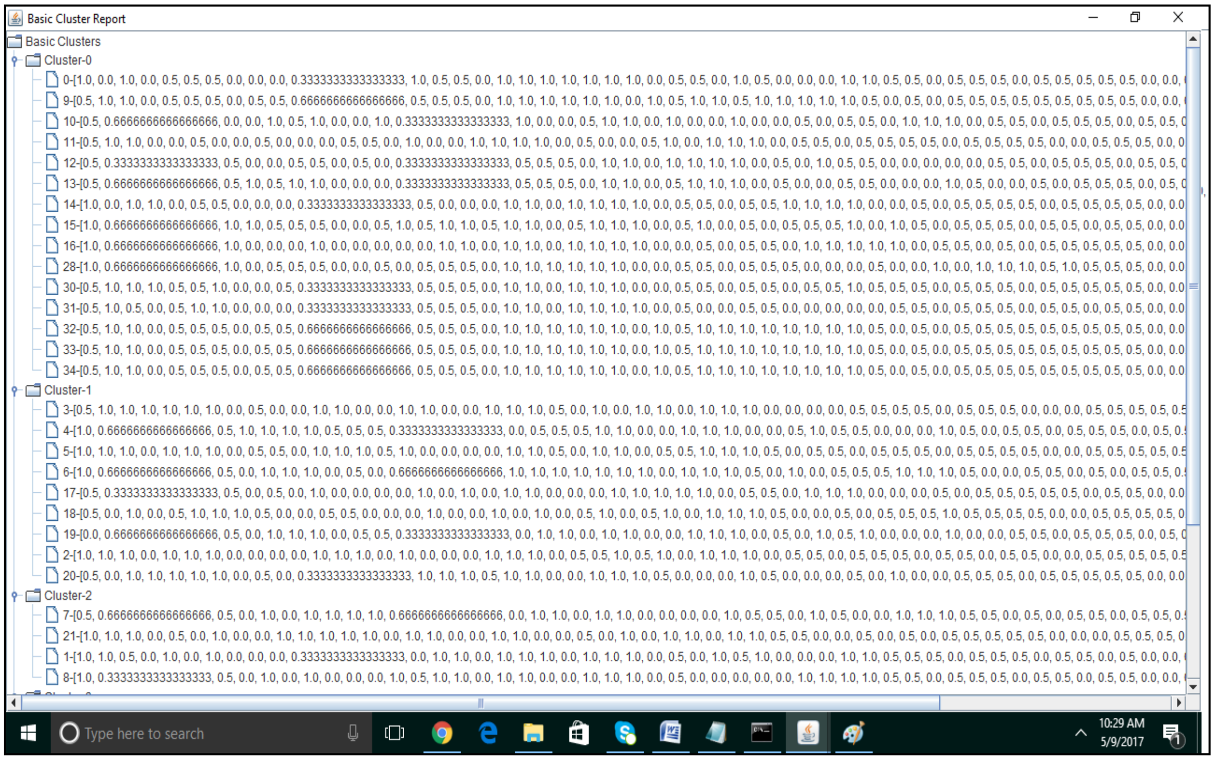
7.5 Final Selected Features(without support count) =42/56

```
FEA-24,0.1667
FEA-19,0.1429
FEA-17,0.2381
FEA-6,0.2381
FEA-25,0.1429
FEA-28,0.1429
FEA-12,0.2381
FEA-49,0.2381
FEA-11,0.119
FEA-35,0.1429
FEA-18,0.2381
FEA-53,0.1429
FEA-4,0.119
FEA-22,0.1905
FEA-5,0.2381
FEA-21,0.2143
FEA-36,0.1429
FEA-27,0.1905
FEA-13,0.119
FEA-23,0.1429
```

Final Selected Features (with support count) =21

```
Total Feature Selection Time=20.627 sec
Nod=20
Total Basic Custers:3
Dimension Limit:50
Similarity Threshold:0.45
Cluster Process Duration (in sec):0.018
CLUSTER-0    R:1.4773408239700376
CLUSTER-1    R:1.4773408239700376
CLUSTER-2    R:3.262264150943396
Basic Cluster Quality:2.0723152662944906
Total Cluster Time=6.922 sec
Total Process Time=27.55sec
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



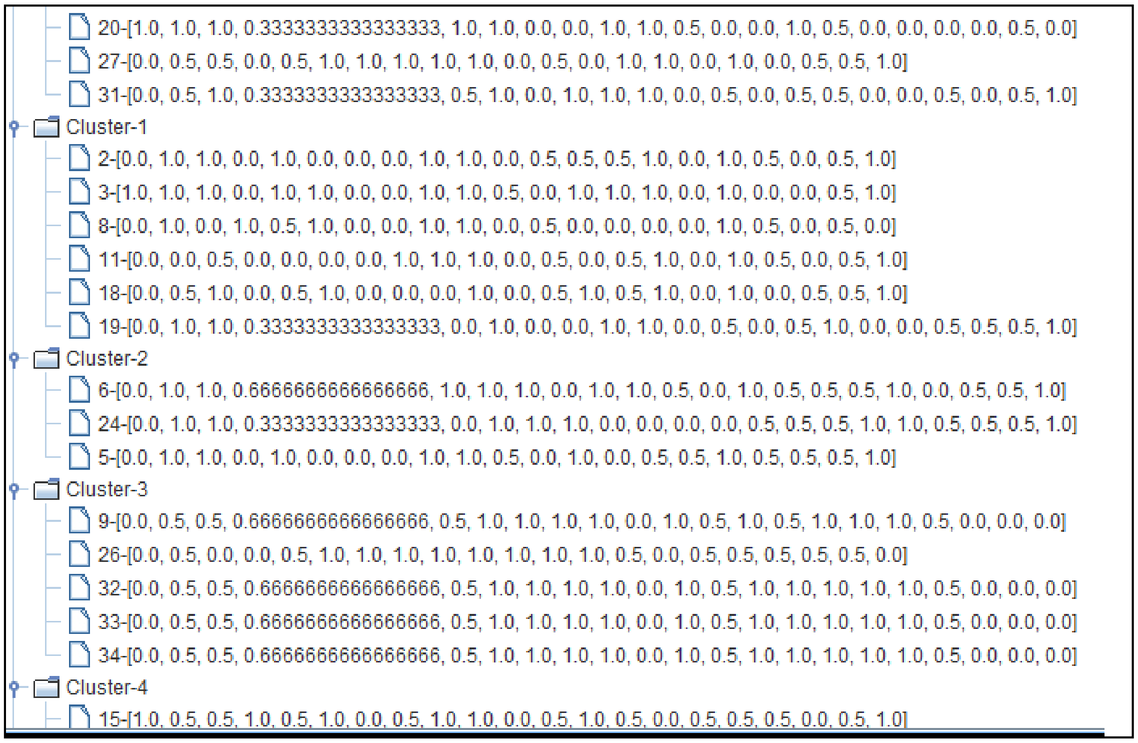
```
Basic Clusters
Cluster-0
  0-[1.0, 0.0, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.3333333333333333, 1.0, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 1.0, 0.5, 0.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  9-[0.5, 1.0, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  10-[0.5, 0.6666666666666666, 0.0, 0.0, 1.0, 0.5, 1.0, 0.0, 0.0, 1.0, 0.3333333333333333, 1.0, 0.0, 0.0, 0.5, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5,
  11-[0.5, 1.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.5, 0.5, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 0.5, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.0,
  12-[0.5, 0.3333333333333333, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.0, 0.3333333333333333, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 1.0, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5,
  13-[0.5, 0.6666666666666666, 0.5, 1.0, 0.5, 1.0, 1.0, 0.0, 0.0, 0.0, 0.3333333333333333, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 0.0, 0.5, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5,
  14-[1.0, 0.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 0.3333333333333333, 0.5, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  15-[1.0, 0.6666666666666666, 1.0, 1.0, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 1.0, 0.5, 1.0, 1.0, 0.5, 1.0, 1.0, 0.0, 0.5, 1.0, 1.0, 1.0, 0.0, 0.5, 1.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 1.0, 0.0, 1.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0,
  16-[1.0, 0.6666666666666666, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  28-[1.0, 0.6666666666666666, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.5, 1.0, 0.5, 0.5, 0.5, 0.0, 0.0,
  30-[0.5, 1.0, 1.0, 1.0, 0.5, 0.5, 1.0, 0.0, 0.0, 0.5, 0.3333333333333333, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  31-[0.5, 1.0, 0.5, 0.0, 0.5, 1.0, 1.0, 0.0, 0.0, 0.0, 0.3333333333333333, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0,
  32-[0.5, 1.0, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  33-[0.5, 1.0, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
  34-[0.5, 1.0, 1.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
Cluster-1
  3-[0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5
  4-[1.0, 0.6666666666666666, 0.5, 1.0, 1.0, 1.0, 1.0, 0.5, 0.5, 0.5, 0.3333333333333333, 0.0, 0.5, 0.5, 0.5, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.5, 0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5
  5-[1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 0.5, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.0, 1.0, 1.0, 0.0, 0.5, 0.5, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5
  6-[1.0, 0.6666666666666666, 0.5, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.6666666666666666, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.0, 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5
  17-[0.5, 0.3333333333333333, 0.5, 0.0, 0.5, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0
  18-[0.5, 0.0, 1.0, 0.0, 0.5, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.5, 1.0, 0.0, 0.5, 1.0, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5,
  19-[0.0, 0.6666666666666666, 0.5, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.3333333333333333, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 1.0, 0.5, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5,
  2-[1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 1.0, 0.5, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5
  20-[0.5, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.3333333333333333, 1.0, 1.0, 1.0, 0.5, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0
Cluster-2
  7-[0.5, 0.6666666666666666, 0.5, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.6666666666666666, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.5, 0.5, 0.0, 1.0, 0.5, 0.0, 0.0, 1.0, 1.0, 1.0, 0.5, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5
  21-[1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.5, 0.5, 0.0, 0.0, 0.5, 0.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5,
  1-[1.0, 1.0, 0.5, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.3333333333333333, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 1.0, 0.5, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.0, 0.0,
  8-[1.0, 0.3333333333333333, 0.5, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.5, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0,
```

Figure 1. Original Data Set With 56 Dimensions

Table 1 Similarity Measure On Each Iteration

Similarity values on each iteration			
0.61	0.57	0.65	0.64
0.57	0.43	0.62	0.36
0.43	0.61	0.51	0.49
0.52	0.57	0.56	0.56
0.35	0.43	0.42	0.42
0.43	0.43	0.47	0.56
0.26	0.52	0.42	0.49
0.52	0.48	0.57	0.60
0.52	0.39	0.65	0.45
0.65	0.48	0.64	0.73
0.65	0.61	0.67	0.73
0.74	0.57	0.71	0.65
0.65	0.48	0.62	0.65
0.65	0.65	0.67	0.65
0.48	0.74	0.52	0.65
0.52	0.57	0.58	0.65

After completion of the similarity measurement the data set dimensions are reduced and shown in to the following figure 3.



```
20-[1.0, 1.0, 1.0, 0.3333333333333333, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.0, 0.0, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.5, 0.0]
27-[0.0, 0.5, 0.5, 0.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.5, 0.5, 1.0]
31-[0.0, 0.5, 1.0, 0.3333333333333333, 0.5, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.5, 0.5, 0.0, 0.0, 0.5, 0.0, 0.5, 1.0]
Cluster-1
2-[0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.5, 0.5, 0.5, 1.0, 0.0, 1.0, 0.5, 0.0, 0.5, 1.0]
3-[1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.5, 1.0]
8-[0.0, 1.0, 0.0, 1.0, 0.5, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.5, 0.0]
11-[0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.5, 1.0, 0.0, 1.0, 0.5, 0.0, 0.5, 1.0]
18-[0.0, 0.5, 1.0, 0.0, 0.5, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.5, 1.0, 0.5, 1.0, 0.0, 1.0, 0.0, 0.5, 0.5, 1.0]
19-[0.0, 1.0, 1.0, 0.3333333333333333, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.5, 1.0, 0.0, 0.0, 0.5, 0.5, 0.5, 1.0]
Cluster-2
6-[0.0, 1.0, 1.0, 0.6666666666666666, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.5, 0.5, 0.5, 1.0, 0.0, 0.5, 0.5, 1.0]
24-[0.0, 1.0, 1.0, 0.3333333333333333, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5, 1.0, 1.0, 0.5, 0.5, 0.5, 1.0]
5-[0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.0, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5, 1.0]
Cluster-3
9-[0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 0.5, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0]
26-[0.0, 0.5, 0.0, 0.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0]
32-[0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0]
33-[0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0]
34-[0.0, 0.5, 0.5, 0.6666666666666666, 0.5, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0]
Cluster-4
15-[1.0, 0.5, 0.5, 1.0, 0.5, 1.0, 0.0, 0.5, 1.0, 1.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 1.0]
```

Figure 3 Dimensions reduced to 21 columns

After reducing the dimensions system asked threshold value for both original data and reduced data, if the threshold value is 0.6 for both it will produce the following clusters and calculate the cluster quality, original data cluster quality is **2.27** and reduced with cluster quality value **2.39**. It will shows on the figure 4. In this implementation cluster quality is increased comparatively existing one, it is not very high difference but this implementation is more use ful for unstructured data. So I believe that this process is better then the existing one. And Mr. Yijun Sun also implemented for feature selection by using I-relief algorithm, in this implementation computation process is very difficult if we implement for very large data sets it will be difficult based on the computational difficulties. In my proposed approach it is not that much of difficult because we are implemented using map reducers it may use it for unstructured data also. The main role of map reducers is distributed the entire data into small small files and distributed to different mappers. If number of mappers is increased automatically time complexity will be reduced.

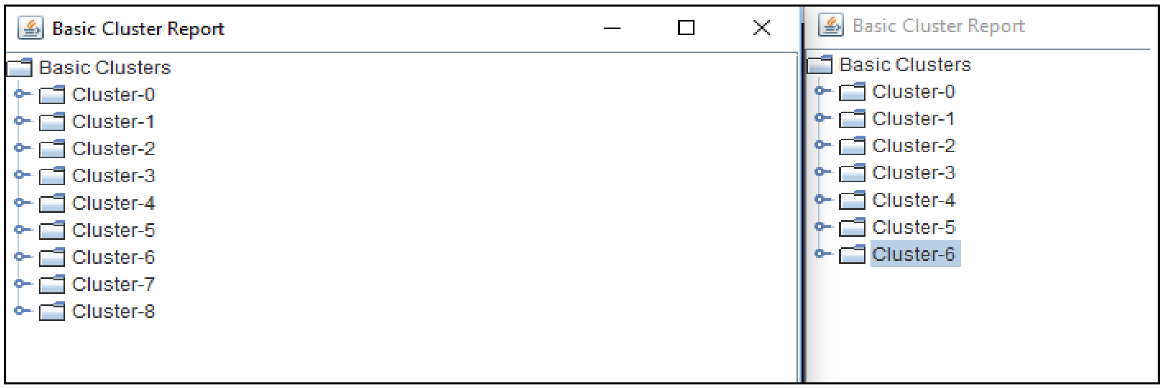


Figure 4. Clusters generated for Reduced Dimensions and Original dimensions

8. CONCLUSION

Based on the existing dimensionality reduction methods map reducer method is also one option to implement and get the better results. Everyone knows Knn is one of the best algorithms to identify nearest neighbors for normal data sets. If we implement along with map reducers it can use it for any type of data sets. In this paper we implement Knn feature selection algorithm to get the better results for high dimensional data it is very simple by using existing java programming language with RMI. If the same apply for very high dimensional data and big data it may not be support but if we increase the number mappers in program it will work for very high dimensional data. It is very simple and useful to implement dimensionality reduction with efficient process.

REFERENCES

[1] Roweis, S. T.; Saul, L. K. (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". *Science*. **290** (5500): 2323 –2326. doi:10.1126science.290.5500.2323 PMID 11125150
[2] D. Koller and M. Sahami, “Toward Optimal Feature Selection,” Proc. 13th Int’l Workshop Machine Learning (ICML ’96), pp. 284-292,1996.
[3] Yijun Sun “Iterative RELIEF for Feature Weighting: Algorithms, Theories, and Applications” IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 6, june 2007.
[4] K. Kira and L.A. Rendell, “A Practical Approach to Feature Selection,” Proc. Ninth Int’l Conf. Machine Learning, pp. 249-256,1992.
[5] T.G. Dietterich, “Machine Learning Research: Four Current Directions,” AI Magazine, vol. 18, no. 4, pp. 97-136, 1997.
[6] https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/
[7] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, “Linear Programming Boosting via Column Generation,” Machine Learning, vol. 46,pp. 225-254, 2002.
[8] Sun and J. Li, “Iterative RELIEF for Feature Weighting,” Proc. 23rd Int’lConf. Machine Learning, pp. 913-920, 2006.
[9] Ali Mustafa Qamar1 and Eric Gaussier2 RELIEF Algorithm and Similarity Learning for k-NN International Journal of Computer Information Systems and Industrial Management Applications.