Data Clustering and Validation For Traffic Management Support System

By

Sadia Nawrin

ID: 2012-2-60-055

Supervised By

Dr. Md. Shamim Akhter

Assistant Professor

Department of Computer Science and Engineering

East West University

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelors of Science in Computer Science and Engineering

to the

Department of Computer Science and Engineering

East West University

Dhaka, Bangladesh

August, 2016

Abstract

Clustering is used to classify or group the sets of data in unsupervised way. Different clustering algorithms including partition k-mean and hierarchical k mean with Elbow method, Davies boulder index, Dunn index, Silhouette coefficient are implemented to find out the optimal classification on the features of traffic management support system. Thereafter, the validity of the optimal classes are analyzed by WSS (within sum of square) errors. Dunn index seems better than others.
Correlation method is used to find the better clustering algorithm between K-mean and hierarchical algorithms. According to correlation partition k-means performs better. Thus, partition k mean with Dunn index is used to determine the optimum number of cluster for five dimensional road weights.

Declaration

I hereby declare that, this project has been done under CSE497 and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication.

Sadia Nawrin
ID: 2012-2-60-055
Department of Computer Science and Engineering
East West University

Letter of Acceptance

We hereby declare that this thesis is from the student's own work and best effort of mine, and all other source of information used have been acknowledge. This thesis has been submitted with our approval.

_____

Dr. Md. Shamim Akhter **Supervisor**

Assistant Professor

Department of Computer Science and Engineering

East West University

_____

Dr. Mozammel Huq Azad Khan **Chairperson**

Professor and Chairperson

Department of Computer Science and Engineering

East West University

Acknowledgement

First of all, I would like to thank almighty Allah for giving me the strength & proper knowledge to complete my thesis work.

I would like to express my deep sense of gratitude and sincere thanks to my honorable supervisor Dr. Md. Shamim Akhter, Assistant Professor, Department of Computer Science and Engineering, East West University, Aftabnagar, Dhaka, Bangladesh for his at most direction, encouragement, kind guidance, sharing knowledge and constant inspiration throughout this thesis work.

My sincere gratefulness for the faculty of Computer Science and Engineering whose friendly attitude and enthusiastic support that has given me for four years.

I am very grateful for the motivation and stimulation from my friends and seniors.

Finally my most heartfelt gratitude goes to my beloved parents and sisters for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to the end.

Abbreviation and Acronyms

HDC=Hierarchical Divisive Clustering

SSE=Sum of Square Error

WSS=Within Sum of Square

SC=Silhouette Coefficient

Dunn=Dunn Index

DB=Davies Bouldin Index

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to traffic jam pollution ten thousand people died every year across the Europe (BBC reported, 2000) [1].Traffic management support system designed for reduce traffic jam. In this system, it collects those feature data from internet which are caused to increase traffic jam. Previously, those feature data are calculated in static way or manually to measure road weight [2]. In this paper we classify those feature data and road weight using clustering. Clustering is the way to classify data in unsupervised technique. To determine the environmental and road status five features are taken which are directly or indirectly related to increase the traffic jam– rain fall, temperature, wind, humidity and peak hour. From web source we are collected 31 days' data of five feature [3].In this paper we classify the features in different clustering algorithm with validation technique and find the optimal cluster for the feature data and optimal class of road weight.

1.1 Objective of the Research

- ✓ Classify the input data of each feature using k mean and hierarchical clustering algorithm (rainfall, temperature, wind, humidity and peak hour)
- ✓ Find the optimal cluster of each feature using different method
- ✓ Compare two clustering algorithm (DIANA and k mean) which one performs better
- ✓ Finally, after getting the optimal class of each feature process the data to find the optimal road weight class.

1.2 Methodology of the Research

A lots of Algorithm used for clustering data- kmean, k-medoids, hierarchical ,fuzzy c mean DBSCAN, optics, etc. In this paper we use only two clustering algorithm k mean and hierarchical. There are lots of method used for determining the optimal cluster such as Elbow method, X-means clustering, Information Criterion Approach, Information Theoretic Approach, Silhouette method,

cross-validation and so on [4].In this paper we use only three mostly used technique - davies bouldin index,dunn index and silhouette coefficient. For verification of optimal cluster we compute within sum of square error (WSS). For cluster performance measuring we use correlation.

1.3 Results of the Research

We implemented both hierarchical and k mean clustering with davies-bouldin index, dunn index and silhouette in c program to identify which clustering algorithm perform better and also determine optimal cluster. Verify the result by WSS and correlation. It shows that k mean with dunn method performs better.

1.4 Thesis Outline

In the next chapter, a short description about clustering and its validation technique. Chapter 3 contains the methodology of our work. Chapter 4 contains the results of our observation of our thesis. Chapter 5 describe summarization of the work and future possible research idea.

Chapter 2

Background Study

2.1 Classification

There are two type of classification- supervised learning and unsupervised learning [13]. When class label of data are known, the classification of those data is called supervised learning. Many different supervised learning (discriminant analysis) methods are available which used for prediction of the outcome of future objects. If data class label are unknown then we can apply unsupervised learning (cluster analysis) method to classify those data. In this paper, collected data set has no class label so we use unsupervised learning method to classify those data.

2.1.1 Clustering

Clustering of data is a method by which grouped large sets of data into clusters of smaller sets of similar data. Clustering is an unsupervised classification, no predefined classes are need. It finds natural grouping of instances in unlabeled data. Main purpose of clustering is identify a finite set of categories, classes or groups in the given dataset. There are many method used to cluster data. Name of the some unsupervised clustering methods are –partitioning, hierarchical, grid-based, density-based clustering etc. The two most popular frequently used methods are hierarchical and partition clustering.

2.1.1.1 Hierarchical Clustering

Hierarchical clustering means construction of a hierarchy of clusters (dendrogram) is merging or splitting clusters.Dendrogram is a tree of nodes representing the clusters. Root contains the whole data set. There are two variant of hierarchical clustering:

a) Agglomerative Hierarchical clustering algorithm (HAC) or AGNES (agglomerative nesting) or bottom-up.
b) Divisive Hierarchical clustering algorithm (HDC) or DIANA (divisive analysis) ortop-down.

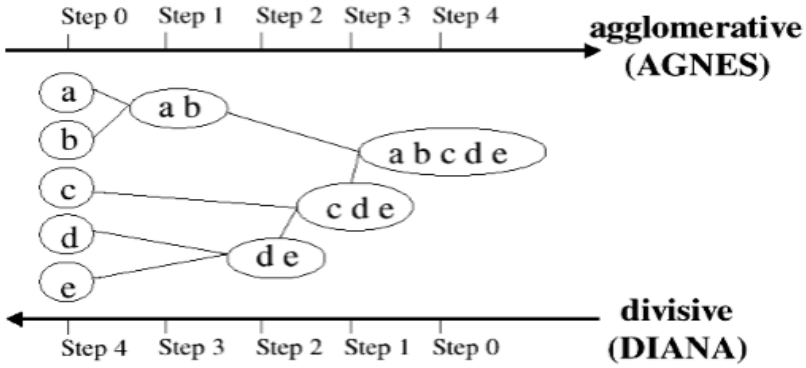

Figure- 2.1: Hierarchical clustering structure.

In this paper, we implemented the divisive hierarchical cluster for classify the feature data.

2.1.1.2 Partitional Clustering

Partitional clustering means determines a flat clustering into k clusters with minimal costs. It partition data set into k cluster. And Assign the object to their nearest center. Here, k is number of centroid.

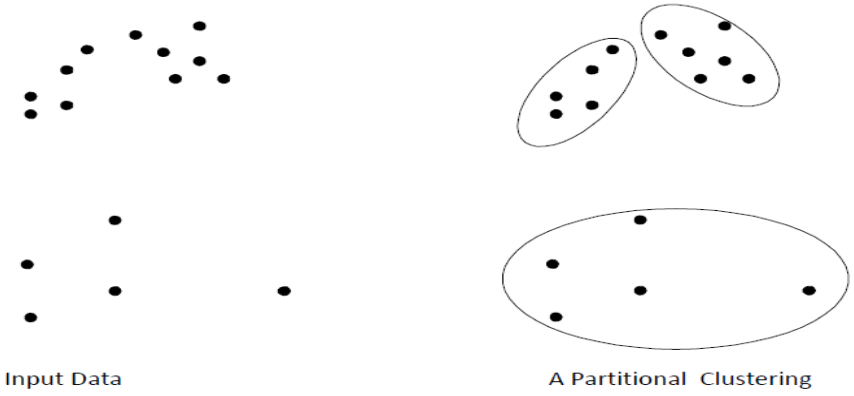

Figure-2.2: Partitional clustering

2.2 Clustering Validation

The procedure of evaluating the results of a clustering algorithm is called clustering validation. It measures goodness of cluster. Clustering validity indexes are usually defined by combining compactness and separability of the clusters. Compactness measures closeness of cluster elements.A common measure of compactness is variance. Separability indicates how distinct two clusters are. There are many aspects of Clustering validation - for compare clustering algorithm, choose clustering parameter. Basically, there are two types of validity technique used for clustering evaluation- external criteria and internal criteria

2.2.1 External Criteria

External criteria is used for categorized data clustering. It compare the clustering results to ground truth. Name of some external criteria clustering validation method are- Jaccard Coefficient, F-measure, Rand index, entropy, purity etc.

2.2.2 Internal Criteria

No internal information is need for internal criteria. Internal criteria is evaluating the quality of clusters use only the data without reference to external information. Some example of internal criteria are davies-bouldin index, dunn index, silhouette coefficient, CH index etc.

2.3 Distance Matrix

Sometimes it called similarity matrix. It is an nxn matrix where n is the number of elements in a data set.d(x, y) distance or dissimilarity between objects x and y.

$$d(x, y) = |x - y|$$

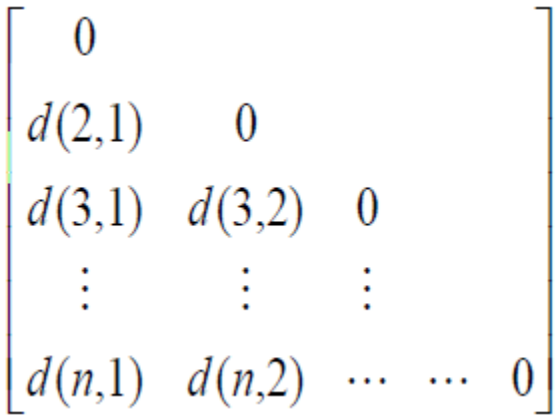

Figure-2.3: Distance matrix or similarity matrix

2.4 Incidence Matrix

An incidence matrix is a matrix that shows the relationship between two classes of objects. It is a nxn matrix where n is the total number of data set. If the object x and the object y belongs to the same cluster then I_{xy}=1 and if the object x and the object y belongs to the different cluster then I_{xy}=0.

Figure-2.4: Incidence matrix

2.5 Manhattan Distance

The absolute distance between the two points. Let, theobjects x = (x1, ..., x_d) and y = (y1, ..., y_d) then the Manhattan distance between two object is,

$$d(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

$$where\ , d = dimention$$

In this thesis, we use manhattan distance measure technique for distance measure.

2.6 Related Work

Clustering was first introduced in anthropology by Driver and Kroeber in 1932. In "Hybridization of the ant colony optimization with the k-means algorithm for clustering", was published by Sara Saatchi, Chih Cheng Hung [17]. In "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values", Zhexue Huang present two algorithms- extend the k-means algorithm to categorical domains and domains with mixed numeric and categorical values and k-modes algorithm uses a simple matching dissimilarity measure to deal with categorical objects [18].

Chapter 3

Materials and Methods

In this chapter, we discuss about algorithms and methods that are used in this paper.
Notation that are used in this paper their description listed below-

Table 3.1: List of symbols and their description used in the mathematical formulation in this paper.

SL No	Symbol/Notation	Description
1.	n_c	Number of total cluster
2.	C_i	i^{th} cluster
3.	d(x,y)	Manhattan distance between two data element
4.	n_i	Number of element in the i^{th} cluster
5.	v_i	Value of the center of the i^{th} cluster
6.	d(v_i,v_j)	Distance between two center
7.	S_i	Variance of i^{th} cluster
8.	C_{kmax}	Maximum number of cluster
9.	d	No of dimension

3.1 Divisive Hierarchical Clustering Algorithm

The Divisive clustering [6] [16] is a variant of hierarchical clustering. Start at the top with all document in one cluster. The cluster is split using flat clustering algorithm (K-mean clustering etc).

3.2.1 Algorithm

a. Initially all items belong to one cluster $C_{i=0}$.
b. Split C_i into sub-cluster, C_{i+1} and C_{i+2}.
c. Apply k mean on C_{i+1} and $C_{i+2.}$
d. Increment the value of i.
e. Repeat steps 2, 3 and 4 until the desired cluster structure is obtained.

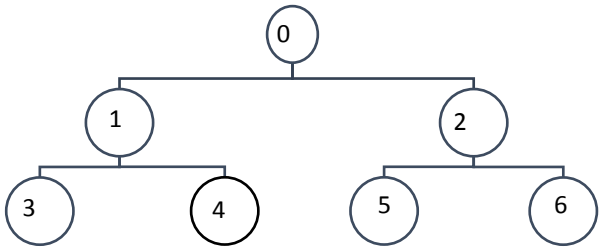

Figure-3.1: Splitting node in DIANA

Node 0 containing the whole data set

C_1=2 input nodes 1, 2.

C_2=3 input nodes-> 2, 3, 4 (1 spilt into 2 sub group-3 and 4).

C_3=4 input nodes ->3,4,5,6 (1 spilt into 2 sub group-5 and 6).

Do until C_{kmax} not reached where C_{kmax} is maximum number of cluster.

3.2 K-mean Clustering

K-means clustering [5] aims to partition data into k clusters in which each data value belongs to the cluster with its nearest cluster.K-means is a most popular nonhierarchical clustering technique. K-means is an iterative algorithm. The basic idea of k-mean is to start with an initial partition and assign objects to clusters so that the squared error decreases.

3.1.1 Algorithm

a. Randomly initialize k center from the set of data point $\{X^d=x_1^d, x_2^d, x_3^d \ldots x_n^d\}$.

b. Assign each point to their nearest center using Manhattan distance measure.

$$\min_{0\le i<n_c} (d(x^d, v_i^d))$$

c. Compute the centroid for each clusterby averaging the data objects belonging to the cluster, assign it as a new cluster center.

$$v_{inew}^d = \frac{1}{n_i}\sum_{k=1}^{n_i} d(x^d, v_{iold}^d)$$

d. Reassign all the data point to its new center.

e. Repeat 2, 3 and 4 until all the cluster centers do not change anymore otherwise stop.

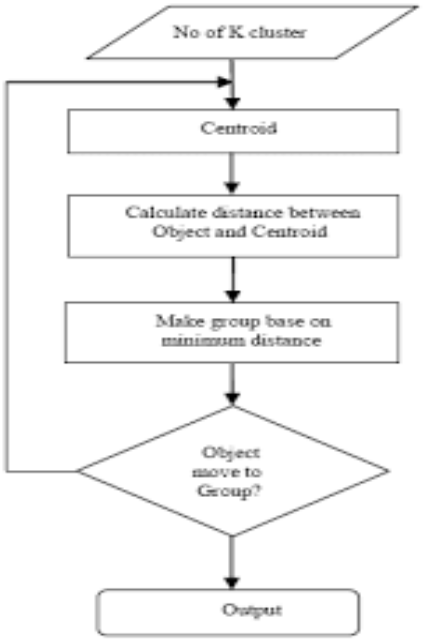

Figure-3-2: Flow chart of k mean algorithm.

3.3 Clustering Validation and Evolution

A good clustering method will produce high quality clusters with high intra-class similarity and low inter-class similarity. There are so many method to measure the quality of the clustering-.the

data that are used for paper has no external information so we used here internal measure or criteria for clustering validation.

3.3.1 Davies-Bouldin Index

Davies Bouldin index [7] [8] measures the average similarity between each cluster and its most similar one. Lower value of Davies Bouldin index indicates that clusters are tight compact and well separated which means better cluster. The goal of this index is achieved minimum within-cluster variance and maximum between-cluster separation.It measure similarity of cluster (R_{ij}) by variance of a cluster (S_i) and separation of cluster (d_{ij}). The formula of DB index is-

$$S_i = \frac{1}{n_i - 1} \sum_{x \epsilon c_i} d(x, v_i)^2$$

$$d_{ij} = d(v_i, v_j)$$

$$R_{ij} = \frac{S_i + S_j}{d_{ij}}$$

$$R_i = \max_{0 \le j < n_c, i \ne j} (R_{ij}) \ , i = 1 \cdots \cdots n_c R_i \ge 0$$

$$DB = \frac{1}{n_c} \sum_{i=1}^{n_c} R_i$$

3.3.3 Dunn Index

The value of Dunn index [9] expected to the large if clusters of the data set are well separated. If the dataset contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the clusters is expected to be small. The clusters that are compact and well separated by maximizing the inter-cluster distance while minimizing the intra-

cluster distance.Large value of dunn index indicate the compact and well-separated clusters. The formula of dunn index is-

$$D = \frac{\min\limits_{0 \le i < n_c, 0 \le j < n_c, i \ne j} (d(C_i, C_j))}{\max\limits_{0 \le k < n_c} (diam(C_i))}$$

Where,

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} (d(x, y))$$

$$diam(C_i) = \max_{x, y \in C_i} (d(x, y))$$

3.3.4 Silhouette Coefficient

Silhouette coefficient [10] [11] shows how objects fit well within the cluster. It measure the quality of the cluster. It is a range between -1and 1-. A value near 1 indicates that the point x is affected to the right cluster.There are two term cohesion and separation. Cohesion is intra clustering distance and separation is distance between cluster centroid. a(x) is the average dissimilarity between x and all other points of its cluster. b(x) is the minimum dissimilarity x and its nearest cluster. Cluster whereas a value near -1 indicates that the point should be affected to another cluster. The formula of silhouette coefficient is-

$$a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i y \ne x} d(x, y)$$

$$b(x) = \min_{j, j \ne i} [\frac{1}{n_j} \sum_{y \in C_j} d(x, y)]$$

$$S = \frac{1}{n_c} \cdot \sum_i \{\frac{1}{n_i} \sum_{x \in C_i} \{\frac{b(x) - a(x)}{\max[b(x), a(x)]}\}\}$$

3.4. Correlation:

An effective clustering algorithm needs a suitable measure of similarity or dissimilarity. Correlation compute the similarity matrix and incident matrix (also called occurrence matrix) to measure the correlation between the data and its cluster [12]. Higher value of correlation indicates that points that belongs to same cluster are very close to each other which means good clustering.The formula ofCorrelation is-

$$r = \frac{\sum_{i=1,j=1}^{n}(d_{ij} - \bar{d})(c_{ij} - \bar{c})}{\sqrt{\sum_{i=1,j=1}^{n}(d_{ij} - \bar{d})^2}\sqrt{\sum_{i=1,j=1}^{n}(c_{ij} - c)^2}}$$

Here,

r =correlation of the data and its cluster,

Distance matrix, D= $\{d_{11},d_{22},d_{33}, \ldots,d_{nn}\}$,

Incident matrix C= $\{c_{11},c_{22},c_{33}, \ldots,c_{nn}\}$,

$\bar{d}$ =mean of the distance matrix,

And $\bar{c}$ =mean of the incident matrix.

4.3.6 Within Sum of Square Error

WSS (Cohesion) measure is also called Sum of Squared Error (SSE) [15]. Sum of square error (SSE) or within sum of square cluster error (WSS) is widely used for criteria measuring. The value of SSE is high indicate high error which means poor quality cluster. Good clustering aims for minimum value of SSE.The formula of within sum of square error is-

$$SSE\ or\ WSS = \sum_{k=1}^{n_c}\sum_{x_i \in C_i} d(x_i, V_i)$$

Chapter 4

Experimental Results and Analysis

A C code implemented based on the above algorithms and methods has been made to determine the optimal feature class and road weight and verified the better algorithm.

4.1 Results of Divisive Hierarchical Algorithm

The sum of square error (SSE) observed of all feature using divisive hierarchical cluster with Davies-Bouldin index, dunn index and silhouette index.

Table 4.1: Optimal number of cluster and value of SSE of each feature using HDC with DB, Dunn and SC method

Feature / Method	Rainfall		temperature		wind		humidity		Peak hour	
	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE
DB	2	25051.32	2	3690.40	2	9301.24	3	31152.06	3	99507.77
Dunn	2	25051.32	2	3690.40	3	4850.62	5	11231.18	4	49786.87
SC	2	25051.32	2	3690.40	3	4850.62	3	31152.06	2	183452.98
Optimal k	2		2		3		5		4	

Shaded block indicates the minimum value of SSE.

Here, the optimal cluster size of each feature using three methods, dunnindex method minimizing the Sum of square error in all cases. So we can say that dunn index better for HDC to find optimal cluster.so the optimal classes of each feature using HDC are – rainfall (K=2), temperature (K=2), wind (K=3), humidity (K=5) and peak hour (K=4).

4.2 Result of K-mean Clustering Algorithm

The sum of square error (SSE) observed of all feature using k-mean cluster with Davies Bouldin index, dunn index and silhouette index.

Table 4.2: Optimal number of cluster and its value of SSE of each feature using k mean with DB, Dunn and SC method

Feature / Method	Rainfall		temperature		wind		humidity		Peak hour	
	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE	Optimal k	SSE
DB	3	9574.97	2	3690.40	2	9301.24	3	23146.38	2	183452.98
Dunn	3	9574.97	3	2106.56	4	4657.67	6	6339.761	5	49541.539
SC	2	25051.32	2	3690.40	2	9301.24	3	23146.38	2	183452.98
Optimal k	3		3		4		6		5	

Shaded block indicates the minimum value of SSE.

Here, also we see that the optimal cluster size of each feature using three methods, dunn index has minimum value of the Sum of square error in all feature. So we can say that dunn index perform better for K mean to find optimal cluster. The optimal classes of each feature using k mean are – rainfall (K=3), temperature (K=3), wind (K=4), humidity (K=6) and peak hour (K=5).

4.3 Comparison of HDC and K-mean Algorithm

In below, hierarchical clustering and k mean clustering are compared by computing the correlation on their optimal cluster for each feature.

Table 4.3: Comparison the correlation between two algorithms

Algorithm / Feature	Hierarchical		K mean	
	Optimal K cluster Using Dunn index	correlation	Optimal K cluster Using Dunn index	correlation
rainfall	2	-0.800799	3	-0.78948
Temperature	2	-0.736153	3	-0.72062
wind	3	-0.579677	4	-0.404638
humidity	5	-0.555058	6	-0.52146
Peak hour	4	-0.638516	5	-0.578275

In table-4.3, we can see that correlations of k mean higher than the correlations of HDC for all feature.so we can say that k mean performs better than HDC.

4.4 Optimal Cluster of Road Weight

Previously, we show that k mean with dunn index perform better for the features. So, for the classification of the road weight we use k mean with dunn index method. Below the table show the no of cluster of road weight and dunn index value of corresponding cluster.

Table 4.4: Cluster size and dun index value of the road weight

No of cluster K	Dunn index value
2	0.08
3	0.09
4	0.10
5	0.11
6	0.11
7	0.13
8	0.12
9	0.12

Here we can see that table 4.4 indicating the max value of dunn index in k=7. So the optimal cluster size of road weight is seven different classes road weight.

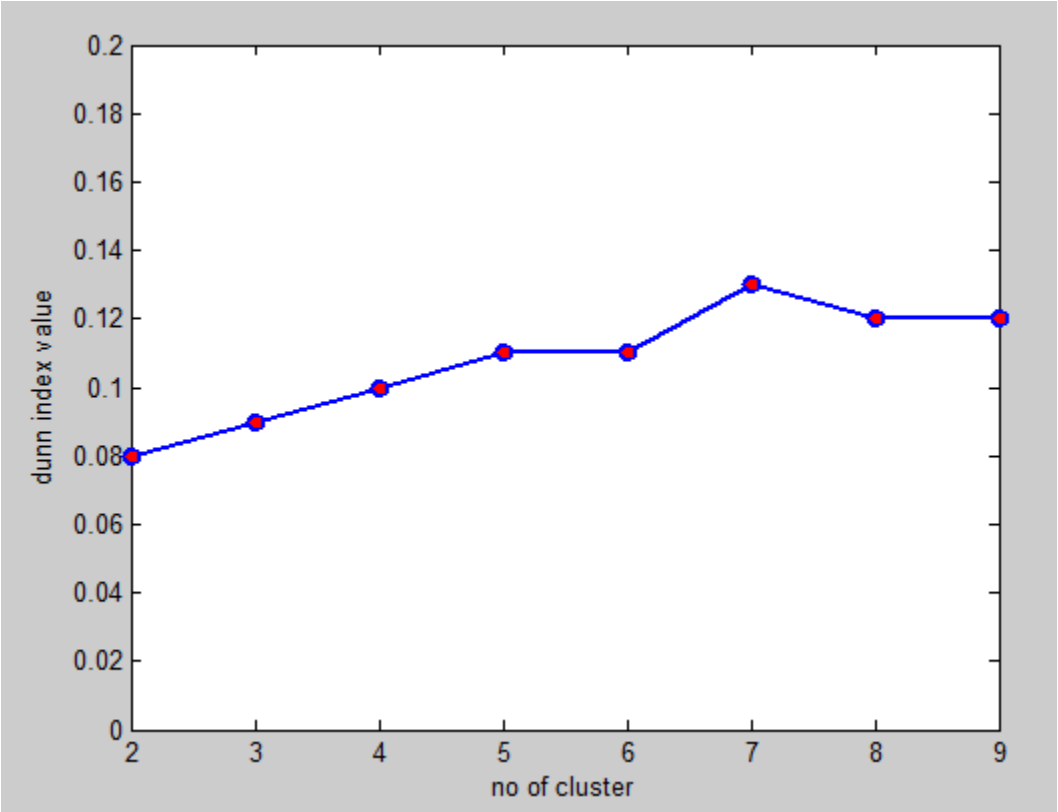

Figure-4.1: No of cluster and dunn index value of k mean (Road weight)

Here, in the above graph indicating the max value of dunn index is pointed on 6 which means k=7. So the optimal cluster size of road weight is sevendifferent classes road weight.

4.5 Sample Clustering Result

Here are some sample of some experimental data-

Table 4.5: Sample road weight clustering result

Data	rainfall	temperature	wind	humidity	peak hour	Road weight
1	0	1	0	3	1	0
2	0	1	0	4	3	5
3	0	0	0	3	2	2
4	0	0	0	3	3	5
5	0	2	1	4	0	4
6	0	2	2	4	1	6
7	0	2	0	3	1	0
8	0	2	1	4	3	3
9	0	2	0	3	0	4
10	0	2	1	5	0	4

In this table-4.5, inputs are the classes of the feature (Rainfall, temperature, wind, humidity and peak hour) and output is the road weight for the feature class values.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this section we summarize our work.The features data are collected from the external feeds (like web site, RSS feed, web service etc.) for classifying data. We cluster the data using two approaches (k-mean and hierarchical). Find optimal number of cluster for those feature data for each approaches using some methods (davies-bouldin index, dunn index and silhouette coefficient).Then compare which algorithm is better for those feature data. After that find the optimal number of cluster of road weight as an input of the clustering is previously measured five feature clusters.

5.2 Future Work

In future, we can also measure validity of the classes by other probabilistic and statistic method.Dunn method needs lots of computational cost. Improve the computation cost and error of the clustering building procedure can be reduce using other statistical model.This Classification need for the neural network classification and prediction.

References

[1] "Traffic Pollution Kills Thousands," BBC News, 2000. Available at: http://news.bbc.co.uk/2/hi/health/905016.stm

[2] Rahman, M.R, and Akhter, S., "Bi-directional traffic management support system with decision tree based dynamic routing", Proc. of 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015, London, United Kingdom, December 14-16, 2015.

[3] https://www.wunderground.com/history/wmo/,17-2-2016.

[4] https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set,3-6-2016.

[5] Reddy, C.K. and Vinzamuri, B., "A Survey of Partitional and Hierarchical Clustering Algorithms", Data Clustering: Algorithms and Applications. CRC Press 2014, ISBN 978-1-46-655821-2 (pg.88-91).

[6] https://sites.google.com/site/dataclusteringalgorithms/hierarchical-clustering-algorithm,15-5-2016.

[7] Kovács, F., Legány, C. and Babos, A., "Cluster Validity Measurement Techniques" ,AIKED'06 Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (2006), ISBN:111-2222-33-9 (pg. 388-393).

[8] Davies, D.L. and Bouldin, D.W. (1979). A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2), 224–227.

[9] Dunn, J.C., "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," J. Cybernetics, vol. 3, 1973, (pg. 32-57).

[10] Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J.,"Understanding of Internal Clustering Validation Measures", Proceeding ICDM '10 Proceedings of the 2010 IEEE International Conference on Data Mining ISBN: 978-0-7695-4256-0 (Pg. 911-916).

[11] Zoubi, M., and Rawi, M., "An efficient approach for computing silhouette coefficients," J ournal of Computer Science 4,(2008) (pg.252–255).

[12] www.cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_basics.pdf, 23-4-2016.

[13] Kincaid, R., "An interactive visualization for exploratory analysis of DNA microarrays", proceeding of the 2004 ACM symposium on applied computing, ISBN: 1-58113-812-1 (pg.167-174).

[14] http://www.academia.edu/3438357/K_Means_Algorithm_Example,12-4-2016

[15] https://hlab.stanford.edu/brian/error_sum_of_squares.html,15-5-2016.

[16] Pelleg, D., Moore, A.W., "X-means: Extending K-means with Efficient Estimation of the Number of Clusters", Proceeding ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning, ISBN: 1-55860-707-2 (Pg.727-734).

[17] Saatchi, S., and Hung, C.C., "Hybridization of the ant colony optimization with the k-means algorithm for clustering", Published in Proceeding SCIA'05 Proceedings of the 14th Scandinavian conference on Image Analysis (Pg. 511-520)

[18] Huang, Z., "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values", Published in Journal Data Mining and Knowledge Discovery archive Volume 2 Issue 3, September 1998 (Pg. 283 – 304)

Appendix:

Code of hierarchical clustering with Davies, Dunn, Silhouette, SSE, correlation

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int ind[1000],set=750;
float inter_dist[5000],intra_dist[5000],si[5000],dist[1000][1000],inc[1000][1000],data[1000];
float points[1000],cluster[100][1000];
float db[10000],dun[10000],sill[10000],vari[10000],sse[10000],cor[10000];
int col_data(int num){
   FILE *fp;
   if(num==0)
      fp=fopen("rainfall-l.txt", "r");
   else if(num==1)
      fp=fopen("temp-l.txt", "r");
   else if(num==2)
      fp=fopen("wind-l.txt", "r");
   else if(num==3)
      fp=fopen("humidity-l.txt", "r");
   else
      fp=fopen("peek_hour-l.txt", "r");
   float jp,tmp;
   int i,n,j;
   i=0;
   while(fscanf(fp, "%f", &jp) != EOF){
         points[i]=data[i]=jp;
         i++;
   }
  n=i;
return n;
}
float absu(float a){
 if(a<0){
   a=a*-1;
 }
 return a;
}
int minimum(float r,int s,int clus){
   int i,w;
   float mini,q;
   mini=10000;
```

```
    for(i=s;i<clus+s;i++){
        q=absu(cluster[i][0]-r);
        if(q<mini){
            mini=q;
            w=i;
        }
    }
    return w;
}
float avg(int index,float clust[]){
    float a;
    int i;
    a=0;
    for(i=1;i<index;i++){
        a=a+clust[i];
    }
    a=a/(index-1);
return a;
}
int min(float r,int s){
    int i,w,clus=2;
    float mini,q;
    mini=10000;
    for(i=s;i<clus+s;i++){
        q=absu(cluster[i][0]-r);
        if(q<mini){
            mini=q;
            w=i;
        }
    }
    return w;
}
void ini(float data[],int dat,int s){
int j,k,r,c,clus=2;
float precenter[100];
k=0;
  for(j=s;j<clus+s;j++){
    for(r=s;r<j;r++){
        if(cluster[r][0]==data[k]){
            k++;
            r=-1;
        }
    }
        precenter[j]=cluster[j][0]=data[k];// printf("\n %f ",data[k]);
        k++;
        ind[j]=1;
    }
```

```
    for(j=0;j<dat;j++){      //assign object closet center
        k=min(data[j],s);
        cluster[k][ind[k]]=data[j]; // printf("\n%f   %d %d",data[j],k,ind[k]);
        ind[k]++;
   }
   for(j=s;j<clus+s;j++){
      cluster[j][0]=avg(ind[j],cluster[j]);
   }
   //printf("\ncc=%f  %f  %d",cluster[s][0],cluster[s+1][0],ind[s+1]);
c=1;
for(r=0;c!=0;r++){          //update center for each cluster if no change then stop iteration
   c=0;
for(j=s;j<clus+s;j++){
      ind[j]=1;
   }
      for(j=0;j<dat;j++){
        k=min(data[j],s);
        cluster[k][ind[k]]=data[j];  // printf("\n%f   %d %d",data[j],k,ind[k]);
        ind[k]++;
   }
   for(j=s;j<clus+s;j++){
      if(precenter[j]!=cluster[j][0]){
        c=1;
      }
   }
   for(j=s;j<clus+s;j++){
      precenter[j]=cluster[j][0];
   }
   for(j=s;j<clus+s;j++){
     cluster[j][0]=avg(ind[j],cluster[j]);
   }
}
}
float davies(int s,int k){
int i,j;
float maxr,result,sum,c[100],r[100][100],R[100];
for(i=s;i<s+k;i++){
   sum=0;
   for(j=1;j<ind[i];j++){
     sum=sum+(pow((cluster[i][0]-cluster[i][j]),2));
   }
   c[i]=sum/(ind[i]-2);
}
for(i=s;i<s+k;i++){
   for(j=s;j<s+k;j++){
      if(i==j){
```

```
            r[i][j]=-1;
            break;
        }
        r[i][j]=(c[i]+c[j])/absu(cluster[i][0]-cluster[j][0]);
    }
}
for(i=s;i<s+k;i++){
    maxr=-1;
    R[i]=-1;
    for(j=s;j<s+k;j++){
        if(maxr<r[i][j])
            maxr=r[i][j];
    }
    R[i]=maxr;
}
result=0;
for(i=s;i<s+k;i++){
    result=result+R[i];
}
result=result/k;
return result;
}
float dunn(int s,int k){
    int i,j,m,n;
    float d[100][100],min,h,max,diam[100],sum;
    for(i=s;i<s+k;i++){
        for(j=s;j<s+k;j++){
            min=10000;
            for(m=1;m<ind[i];m++){
                for(n=1;n<ind[j];n++){
                        h=absu(cluster[i][m]-cluster[j][n]);
                        if(h<min){
                            min=h;
                        }
                }
            }
            d[i][j]=min;
        }
    }
for(i=s;i<s+k;i++){
        max=-1;
    for(m=1;m<ind[i];m++){
            for(n=1;n<ind[i];n++){
            h=absu(cluster[i][m]-cluster[i][n]);
            if(h>max){
                max=h;
            }
```

```
        }
    }
    diam[i]=max;
}
     max=-1;
    for(i=s;i<s+k;i++){
        if(diam[i]>max)
            max=diam[i];
    }
    sum=0;
    min=1000;
for(i=s;i<s+k;i++){

    for(j=s;j<s+k;j++){
        if(d[i][j]<min&&i!=j){
            min=d[i][j];
        }
    }
}
    h=min/max;
return h;
}
float sillhouette(int st,int s){
    float mini,maxi,sum;
    int i,j,m,k;
    float avg;
    for(i=st;i<st+s;i++){
        si[i]=0;
    }
    for(i=0;i<set;i++){
        k=minimum(data[i],st,s);
        avg=0;
        for(j=1;j<ind[k];j++){
            avg=avg+absu(cluster[k][j]-data[i]);
        }
        if(ind[k]>1)
            avg=avg/(ind[k]-2);
        else
            avg=0;
        intra_dist[i]=avg;
        mini=100000;
        for(j=st;j<st+s;j++){
            if(j!=k){
                avg=0;
                for(m=1;m<ind[j];m++){
                    avg=avg+absu(cluster[j][m]-data[i]);
                }
```

```
            avg=avg/(ind[j]-1);
            if(mini>avg)
               mini=avg;
               }
       }
       inter_dist[i]=mini;
       if(inter_dist[i]<intra_dist[i]){
            maxi=intra_dist[i];
          }
          else{
            maxi=inter_dist[i];
          }
          si[k]=si[k]+((inter_dist[i]-intra_dist[i])/maxi);
    }
    sum=0;
   for(i=st;i<st+s;i++){
       sum=sum+(si[i]/(ind[i]-1));
    }
   sum=sum/s;
    return sum;
}
float err(int s,int k){
float result=0;
int i,j;
for(i=s;i<s+k;i++){
    for(j=1;j<ind[i];j++){
     result=result+(pow((cluster[i][0]-cluster[i][j]),2));
    }
}
return result;
}
float correlation(int s,int k){
    float result,avg_inc,avg_dist,eq_d,eq_i;
    int i,j,m,in[5000];
    for(m=0;m<set;m++){
        in[m]=minimum(data[m],s,k);
       }
       for(i=0;i<set;i++){
          for(j=0;j<set;j++){
             dist[i][j]=absu(data[i]-data[j]);
              if(in[i]==in[j])
             inc[i][j]=1;
          else
             inc[i][j]=0;
          }
       }
       avg_dist=avg_inc=0;
```

```
        for(i=0;i<set;i++){
            for(j=0;j<i;j++){
                avg_dist=avg_dist+dist[i][j];
                avg_inc=avg_inc+inc[i][j];
            }
        }
        avg_dist=avg_dist/((set*(set-1))/2);
        avg_inc=avg_inc/((set*(set-1))/2);
        result=eq_i=eq_d=0;
        for(i=0;i<set;i++){
            for(j=0;j<i;j++){
                eq_d=eq_d+pow((dist[i][j]-avg_dist),2);
                eq_i=eq_i+pow((inc[i][j]-avg_inc),2);
            }
        }
        eq_d=pow(eq_d,.5);
        eq_i=pow(eq_i,.5);
        for(i=0;i<set;i++){
            for(j=0;j<i;j++){
                result=result+(((dist[i][j]-avg_dist)*(inc[i][j]-avg_inc)));
            }
        }
        result=result/(eq_d*eq_i);
    return result;
}
int main(){
int point_count,i,s,k_max=11,n,num;
float sum;
FILE *fp1;
for(num=0;num<5;num++){
    if(num==0){
    fp1=fopen("hierarchical-rainfall.txt", "w");
    printf("\nrainfall\n");
    printf("--------");
    }
    else if(num==1){
    fp1=fopen("hierarchical-temperature.txt", "w");
    printf("\ntemperature\n");
    printf("-----------");
    }
    else if(num==2){
        fp1=fopen("hierarchical-wind.txt", "w");
        printf("\nwind\n");
        printf("----");
    }
    else if(num==3){
        fp1=fopen("hierarchical-humidity.txt", "w");
```

```
        printf("\nhumidity\n");
        printf("--------");
    }
    else{
        fp1=fopen("hierarchical-peek_hour.txt", "w");
        printf("\npeek_hour\n");
        printf("--------");
    }
    point_count=col_data(num);
sum=0;
for(i=0;i<point_count;i++){
    sum=sum+points[i];
    cluster[0][i+1]=points[i];
}
ind[0]=point_count+1;
sum=sum/point_count;
cluster[0][0]=sum;
ini(points,point_count,1);
for(n=1;n<k_max;n++){            //hierarchical 2 mean
    for(i=1;i<ind[n];i++){
        points[i-1]=cluster[n][i];
    }
ini(points,ind[n]-1,n*2+1);
}
for(i=2;i<=k_max;i++){
    db[i]=davies(i-1,i);
    //printf("davies-bouldin-%d=%f\n",i,db[i]);
    dun[i]=dunn(i-1,i);
   // printf("dunn index-%d=%f\n",i,dun[i]);
    sill[i]=sillhouette(i-1,i);
    //printf("sillhouette-%d=%f\n",i,sill[i]);*/
    cor[i]=correlation(i-1,i);
   // printf("correlation-%d=%f\n",i,cor[i]);
    sse[i]=err(i-1,i);
    //printf("sum of square error-%d=%f\n\n",i,sse[i]);
}
return 0;
}
```

Code of K mean clustering with Davies Bouldin, Dunn index, Silhouette, SSE, Correlation

```
#include<stdio.h>
#include<math.h>
static int dat=750,kmax=11;
float db[10000],dun[10000],sill[10000],vari[10000],sse[10000],cor[10000];
float inter_dist[5000],intra_dist[5000],si[5000],dist[1000][1000],inc[1000][1000];
float data[1000];
float cluster[100][1000],precenter[1000];
int ind[100],c;
void col_data(int num){
   FILE *fp;
   if(num==0)
      fp=fopen("rainfall-l.txt", "r");
   else if(num==1)
      fp=fopen("temp-l.txt", "r");
   else if(num==2)
      fp=fopen("wind-l.txt", "r");
   else if(num==3)
      fp=fopen("humidity-l.txt", "r");
   else
      fp=fopen("peek_hour-l.txt", "r");
   float j;
   int i;
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i]=j;
        i++;
   }
}
float absu(float a){
  if(a<0){
    a=a*-1;
  }
  return a;
  }
int min(float r,int clus){
   int i,w;
   float mini,q;
   mini=10000;
   for(i=0;i<clus;i++){
      q=absu(cluster[i][0]-r);
      if(q<mini){
         mini=q;
```

```
            w=i;
        }
    }
    return w;
}
float avg(int s){
    float a;
    int i;
    a=0;
    for(i=1;i<ind[s];i++){
        a=a+cluster[s][i];
    }
    a=a/(ind[s]-1);
return a;
}
void ini(int clus){
int j,k,r;
k=0;
    for(j=0;j<clus;j++){
    for(r=0;r<j;r++){
        if(cluster[r][0]==data[k]){
            k++;
            r=-1;
        }
    }
        precenter[j]=cluster[j][0]=data[k]; //printf("\n %f ",data[k]);
        k++;
        ind[j]=1;
    }
        for(j=0;j<dat;j++){      //assign object closet center
            k=min(data[j],clus);
            cluster[k][ind[k]]=data[j]; // printf("\n%f   %d %d",data[j],k,ind[k]);
            ind[k]++;
    }
    for(j=0;j<clus;j++){
        cluster[j][0]=avg(j);
    }
}
void kmean(int clus){
     int j,k,r,i;
     float mini,maxi;
    c=1;
for(r=1;c!=0;r++){          //update center for each cluster if no change then stop iteration
    c=0;
for(j=0;j<clus;j++){
        ind[j]=1;
    }
```

```
        for(j=0;j<dat;j++){
            k=min(data[j],clus);
            cluster[k][ind[k]]=data[j];
            ind[k]++;
    }
    for(j=0;j<clus;j++){
        if(precenter[j]!=cluster[j][0]){
            c=1;
 }}
    for(j=0;j<clus;j++){
        precenter[j]=cluster[j][0];
    }
    for(j=0;j<clus;j++){
      cluster[j][0]=avg(j);
    }
}
float davies(int k){
int i,j;
float maxr,result,sum,c[100],r[100][100],R[100];
for(i=0;i<k;i++){
    sum=0;
    for(j=1;j<ind[i];j++){
      sum=sum+(pow((cluster[i][0]-cluster[i][j]),2));
    }
    c[i]=sum/(ind[i]-2);
}
for(i=0;i<k;i++){
    for(j=0;j<k;j++){
        if(i==j){
            r[i][j]=-1;
            break;
        }
        r[i][j]=(c[i]+c[j])/absu(cluster[i][0]-cluster[j][0]);
    }}
for(i=0;i<k;i++){
    maxr=-1;
    R[i]=-1;
    for(j=0;j<k;j++){
        if(maxr<r[i][j])
            maxr=r[i][j];
    }
    R[i]=maxr;
}
result=0;
for(i=0;i<k;i++){
    result=result+R[i];
}
```

```
result=result/k;
return result;
}
float dunn(int k){
   int i,j,m,n;
   float d[100][100],min,h,max,diam[100],sum;
   for(i=0;i<k;i++){
      for(j=0;j<k;j++){
         min=10000;
         for(m=1;m<ind[i];m++){
            for(n=1;n<ind[j];n++){
                  h=absu(cluster[i][m]-cluster[j][n]);
                  if(h<min){
                     min=h;
                  } }
         }
         d[i][j]=min;
      }
   }
for(i=0;i<k;i++){
      max=-1;
   for(m=1;m<ind[i];m++){
         for(n=1;n<ind[i];n++){
         h=absu(cluster[i][m]-cluster[i][n]);
         if(h>max){
            max=h;
         }
      }
   }
   diam[i]=max;
}

   max=-1;
   for(i=0;i<k;i++){
      if(diam[i]>max)
         max=diam[i];
   }
   sum=0;
   min=1000;
for(i=0;i<k;i++){
   for(j=0;j<k;j++){
      if(d[i][j]<min&&i!=j){
         min=d[i][j];
      }
   }
}
   h=min/max;
```

```
return h;
}
float correlation(int k){
   float result,avg_inc,avg_dist,eq_d,eq_i;
   int i,j,m,in[5000];
   for(m=0;m<dat;m++){
       in[m]=min(data[m],k);
      }
      for(i=0;i<dat;i++){
         for(j=0;j<dat;j++){
            dist[i][j]=absu(data[i]-data[j]);
             if(in[i]==in[j])
            inc[i][j]=1;

         else
            inc[i][j]=0;
         }
      }
      avg_dist=avg_inc=0;
      for(i=0;i<dat;i++){
         for(j=0;j<i;j++){
            avg_dist=avg_dist+dist[i][j];
            avg_inc=avg_inc+inc[i][j];
        }
      }
      avg_dist=avg_dist/((dat*(dat-1))/2);
      avg_inc=avg_inc/((dat*(dat-1))/2);
      result=eq_i=eq_d=0;
      for(i=0;i<dat;i++){
         for(j=0;j<i;j++){
            eq_d=eq_d+pow((dist[i][j]-avg_dist),2);
            eq_i=eq_i+pow((inc[i][j]-avg_inc),2);
         }
      }
      eq_d=pow(eq_d,.5);
      eq_i=pow(eq_i,.5);
      for(i=0;i<dat;i++){
         for(j=0;j<i;j++){
            result=result+(((dist[i][j]-avg_dist)*(inc[i][j]-avg_inc)));
         }
      }
      result=result/(eq_d*eq_i);
   return result;
}
float sillhouette(int s){
   float mini,maxi,sum;
   int i,j,m,k;
```

```
    float avg;
    for(i=0;i<s;i++){
        si[i]=0;
    }
    for(i=0;i<dat;i++){
        k=min(data[i],s);
        avg=0;
        for(j=1;j<ind[k];j++){
            avg=avg+absu(cluster[k][j]-data[i]);
        }
        if(ind[k]>1)
            avg=avg/(ind[k]-2);
        else
            avg=0;
        intra_dist[i]=avg;
        mini=100000;
        for(j=0;j<s;j++){
            if(j!=k){
                avg=0;
                for(m=1;m<ind[j];m++){
                    avg=avg+absu(cluster[j][m]-data[i]);
                }
                avg=avg/(ind[j]-1);
                if(mini>avg)
                    mini=avg;
                    }
        }
        inter_dist[i]=mini;
            if(inter_dist[i]<intra_dist[i]){
                maxi=intra_dist[i];
            }
            else{
                maxi=inter_dist[i];
            }
            si[k]=si[k]+((inter_dist[i]-intra_dist[i])/maxi);
    }
    sum=0;
    for(i=0;i<s;i++){
        sum=sum+(si[i]/(ind[i]-1));
    }
      sum=sum/s;
    return sum;
}
float err(int k){
float result=0;
int i,j;
for(i=0;i<k;i++){
```

```
    for(j=1;j<ind[i];j++){
     result=result+(pow((cluster[i][0]-cluster[i][j]),2));
    }
}
return result;
}
int main(){
int i,m,j,num;
FILE *fp1;
for(num=0;num<5;num++){
 col_data(num);
    if(num==0){
    fp1=fopen("kmean-rainfall.txt", "w");
    printf("\nrainfall\n");
    printf("--------");
    }
    else if(num==1){
        fp1=fopen("kmean-temperature.txt", "w");
    printf("\ntemperature\n");
    printf("-----------");
    }
    else if(num==2){
        fp1=fopen("kmean-wind.txt", "w");
        printf("\nwind\n");
        printf("----");
    }
    else if(num==3){
        fp1=fopen("kmean-humidity.txt", "w");
        printf("\nhumidity\n");
        printf("--------");
    }
    else{
        fp1=fopen("kmean-peek_hour.txt", "w");
        printf("\npeek_hour\n");
        printf("--------");
    }
 for(i=2;i<=kmax;i++){
 ini(i);
 kmean(i);
 db[i]=davies(i);
 dun[i]=dunn(i);
 sill[i]=sillhouette(i);
 cor[i]=correlation(i);
 sse[i]=err(i);
 }
 return 0;
}
```

Code of Dimensional k mean with Dunn index method

```
#include<stdio.h>
#include<math.h>
static int dat=750,kmax=10,dim=5;
float data[100000][10],precenter[100000][10],maxi[10000],mini[10000];
float cluster[1000][10000][10],dun[100];
int ind[10000],c;
void col_data(){
   FILE *fp;
   fp=fopen("kmean-rainfall.txt", "r");
   float j;
   int i;
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i][0]=j;

        i++;
   }
   fp=fopen("kmean-temperature.txt", "r");
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i][1]=j;
        i++;
   }
   fp=fopen("kmean-wind.txt", "r");
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i][2]=j;

        i++;
   }
   fp=fopen("kmean-humidity.txt", "r");
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i][3]=j;
        i++;
   }
   fp=fopen("kmean-peek_hour.txt", "r");
   i=0;
   while(fscanf(fp, "%f", &j) != EOF){
        data[i][4]=j;
        i++;
   }
}
float avg(int s,int d){
```

```
    float a;
    int i;
    a=0;
    for(i=0;i<ind[s];i++){
       a=a+cluster[s][i][d];
    }
    a=a/(ind[s]-1);
return a;
}

float absu(float a){
  if(a<0){
    a=a*-1;
  }
  return a;
  }
int min(float r[],int clus){
    int i,w,d;
    float mini,q;
    mini=100000;
    for(i=0;i<clus;i++){
       q=0;
     for(d=0;d<dim;d++){
       q=q+absu(cluster[i][0][d]-r[d]);
     }
       if(q<mini){
          mini=q;
          w=i;
       }

    }
    return w;
}
void ini(int clus){
int j,k,r,d,i,flag;
k=0;

    for(j=0;j<clus;j++){
      for(r=0;r<j;r++){
          flag=0;
        for(i=0;i<dim;i++){
          if(cluster[r][0][i]==data[k][i]){
             flag=flag+1;
          }
        }
       if(flag==dim){
          k++;
```

```
            r=-1;
    }
}
 for(d=0;d<dim;d++){
        precenter[j][d]=cluster[j][0][d]=data[k][d]; //printf("\n %f ",data[k][d]);
 }
        ind[j]=1;
}

        for(j=0;j<dat;j++){      //assign object closet center
            k=min(data[j],clus);
            for(d=0;d<dim;d++){
            cluster[k][ind[k]][d]=data[j][d];
            }
            ind[k]++;
    }
    for(j=0;j<clus;j++){
        for(d=0;d<dim;d++){
        cluster[j][0][d]=avg(j,d);
    }
    }



}
void kmean(int clus){
     int j,k,r,d,i,m,flag;

    c=1;
for(r=0;c!=0;r++){           //update center for each cluster if no change then stop iteration
    c=0;

   for(j=0;j<clus;j++){
        ind[j]=1;
    }


        for(j=0;j<dat;j++){
            k=min(data[j],clus);
            for(d=0;d<dim;d++){
            cluster[k][ind[k]][d]=data[j][d]; //printf("\n%f   %d %d",data[j][d],k,ind[k][d]);
            }
            ind[k]++;
            }

    for(j=0;j<clus;j++){
        flag=0;
```

```
        for(d=0;d<dim;d++){
            if(cluster[j][0][d]!=precenter[j][d]){
                flag=flag+1;
            }
        }
        if(flag==dim){
            c=1;
        }

    }

    for(j=0;j<clus;j++){
     for(d=0;d<dim;d++){
        precenter[j][d]=cluster[j][0][d];
    }
    }
    for(j=0;j<clus;j++){
     for(d=0;d<dim;d++){
       cluster[j][0][d]=avg(j,d);
    }
    }
}
}
float dunn(int k){
    int i,j,m,n,dd;
    float d[100][100],min,h,max,diam[100],sum;
    for(i=0;i<k;i++){
        for(j=0;j<k;j++){
            min=10000;
            for(m=1;m<ind[i];m++){
                for(n=1;n<ind[j];n++){
                        h=0;
                        for(dd=0;dd<dim;dd++){
                        h=h+absu(cluster[i][m][dd]-cluster[j][n][dd]);

                }
                if(h<min){
                min=h;
                }
            }
            d[i][j]=min;
        }
    }
}
for(i=0;i<k;i++){
        max=-1;
    for(m=1;m<ind[i];m++){
```

```
            for(n=1;n<ind[i];n++){
               h=0;
               for(dd=0;dd<dim;dd++){
                  h=h+absu(cluster[i][m][dd]-cluster[i][n][dd]);
                  }
            if(h>max){
               max=h;
            }

         }
      }
      diam[i]=max;
}
      max=-1;
      for(i=0;i<k;i++){
         if(diam[i]>max)
            max=diam[i];
      }
      sum=0;
      min=1000;
for(i=0;i<k;i++){
      for(j=0;j<k;j++){
         if(d[i][j]<min&&i!=j){
            min=d[i][j];
         }
      }
}
      h=min/max;
return h;
}
int main(){
int i,j,m,d;
float t;
FILE *fp1;
col_data();
fp1=fopen("kmean-overall.txt", "w");
for(i=2;i<10;i++){
ini(i);
kmean(i);
dun[i]=dunn(i);
fprintf(fp1,"i=%d  %f\n",i,dun[i]);
printf("ncluster is k=%d==%f\n",i,dun[i]);
}
return 0;
}
```