An Efficient Technique for Clustering Data with Mixed Attribute Types

Rahmah Brnawy

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science

Concordia University

Montreal, Quebec, Canada

June 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Rahmah Brnawy**

Entitled: **An Efficient Technique for Clustering Data with Mixed Attribute Types**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________________ Chair
Dr. Peter Rigby

______________________________________ Examiner
Dr. Sudhir Mudur

______________________________________ Examiner
Dr. Gösta Grahne

______________________________________ Supervisor
Dr. Nematollaah Shiri

Approved by ______________________________________ Chair of Department or Graduate Program Director

______________________________________ Dean of Faculty

Date ______________________________________

ABSTRACT

An Efficient Technique for Clustering Data with Mixed Attribute Types

Rahmah Brnawy

Clustering is a technique used to extract useful information and discover patterns from data. Existing clustering techniques have often focused on datasets with attributes that are either numeric or categorical but not both. The problem of clustering mixed numeric and categorical datasets has received increased attention more recently and a number of solutions have been proposed. In this research, we study these solutions and propose two clustering algorithms. The first algorithm that we present is called *Cluc+*, which extends and improves Cluc, an existing algorithm proposed for clustering pure categorical data. Using *Cluc+*, we then develop a new algorithm, called *k-mixed* for clustering data with mixed numeric and categorical attribute types. We conduct numerous experiments to evaluate the performance of our proposed algorithms using real-life benchmark datasets. Our results indicate increased efficiency and accuracy of the proposed solution techniques.

ACKNOWLEDGMENTS

I am grateful to the Almighty God for helping me at all times including through the period of my study at Concordia University. I so much appreciate my supervisor, Dr. Nematollaah Shiri, for his patience and guidance. I cannot forget his encouragements through the learning curve of my research.

I would also like to extend my sincere gratitude to the Saudi Bureau and my alma mater –Tabiah University– for their financial support.

To my wonderful family: my Dad, Mom and siblings, I say thank you so much for all your love and care.

Also, special thanks to my lovely sister, Ebtesam Barnawi, and to my best friend, Mr. Taiwo Adetiloye, who is currently pursuing his PhD study at CIISE, who made my stay at Concordia worthwhile.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

1.1. Cluster Analysis

A huge amount of data is produced and collected every day in different applications. Different mining techniques have been developed and deployed to extract useful information and discover patterns in the data using statistical techniques to obtain, e.g., mean, standard deviation, etc. However, such data often includes other hidden patterns and useful information that are far more valuable to find. This information and knowledge can assist in different application domains such as decisions making, medical diagnosis, business management, etc. Data analysis and mining tools are designed to discover such interesting knowledge and patterns from enormous amount of data.

Clustering is an important data-mining technique, which falls in the category of *unsupervised* learning techniques, as it does not use prior knowledge about the data and its structure [1]. The goal of clustering is to divide the input set of data/objects into homogenous groups (called clusters), such that objects within the same cluster are “similar” to each other and those placed in different clusters are “dissimilar” [1].

In this research, clustering structured data is considered, that is, every data item/object is described by a fixed set of attributes as in relational data. The types of data could be numeric or categorical, i.e., enumerated type. While clustering of numeric data enjoys a number of popular similarity measures such as Euclidean distance, clustering categorical data is often a challenge since there is no inherent distance measure between categorical values [2]. We refer to datasets described by attributes of the same type as *pure* datasets. The attributes of a pure dataset are either all of numeric type or they are all categorical. The class of pure datasets are then classified to pure numeric or pure categorical. Some datasets have a mix of numeric and categorical attributes.

There are numerous solutions proposed for clustering pure numeric data, such as *k-mean* [3], *DBSCAN* [4], *CHAMELEON* [5], and pure categorical data such as *CURE* [6], *ROCK* [7], *CLUC* [8]. Clustering data with mixed data types has received increased attention more recently.

Researchers have been looking for "suitable" ways to determine the "weights" of different attribute types to avoid biased results. Another problem that affects the performance of clustering data with mixed attribute types is that they often treat nominal and binary attributes equally. The previously proposed solutions paid less attention to the fact that binary attributes could be symmetric or asymmetric [2], and hence should not be given the same weights in the clustering process. This implies that a proper assignment of weights is crucial for a desired clustering algorithm to capture the different characteristics of the data.

Similarity measures are important for the performance of clustering pure or mixed data. Consider Figure 1 that was generated by Weka [9] which shows data distribution of a real-life dataset. As can be seen, the dataset contains 9 categorical attributes, of which C, D, E, F, and M are nominal with different distinct values, and A, H, I, and K are binary with two possible values. The distinct values of each attribute are represented by solid rectangles associated with a number on the top, which reflects the frequencies of the values in that attribute. For example, attribute "A" has two different values, therefore there are two sold rectangles. The frequency of one of its value is 488 and the other one is 222. The clustering of this dataset is influenced by the binary attributes, since one of the two values appears most frequently but carries less weight compared to the other value and other nominal values [2]. Since the similarity measures used in clustering categorical data are usually based on the co-occurrences and frequencies of the values, the high frequency of some values may lead to bias in the clustering result and poor quality in general. This suggests that, to improve the clustering quality, we need to reduce the influence of attributes with highly frequent values.

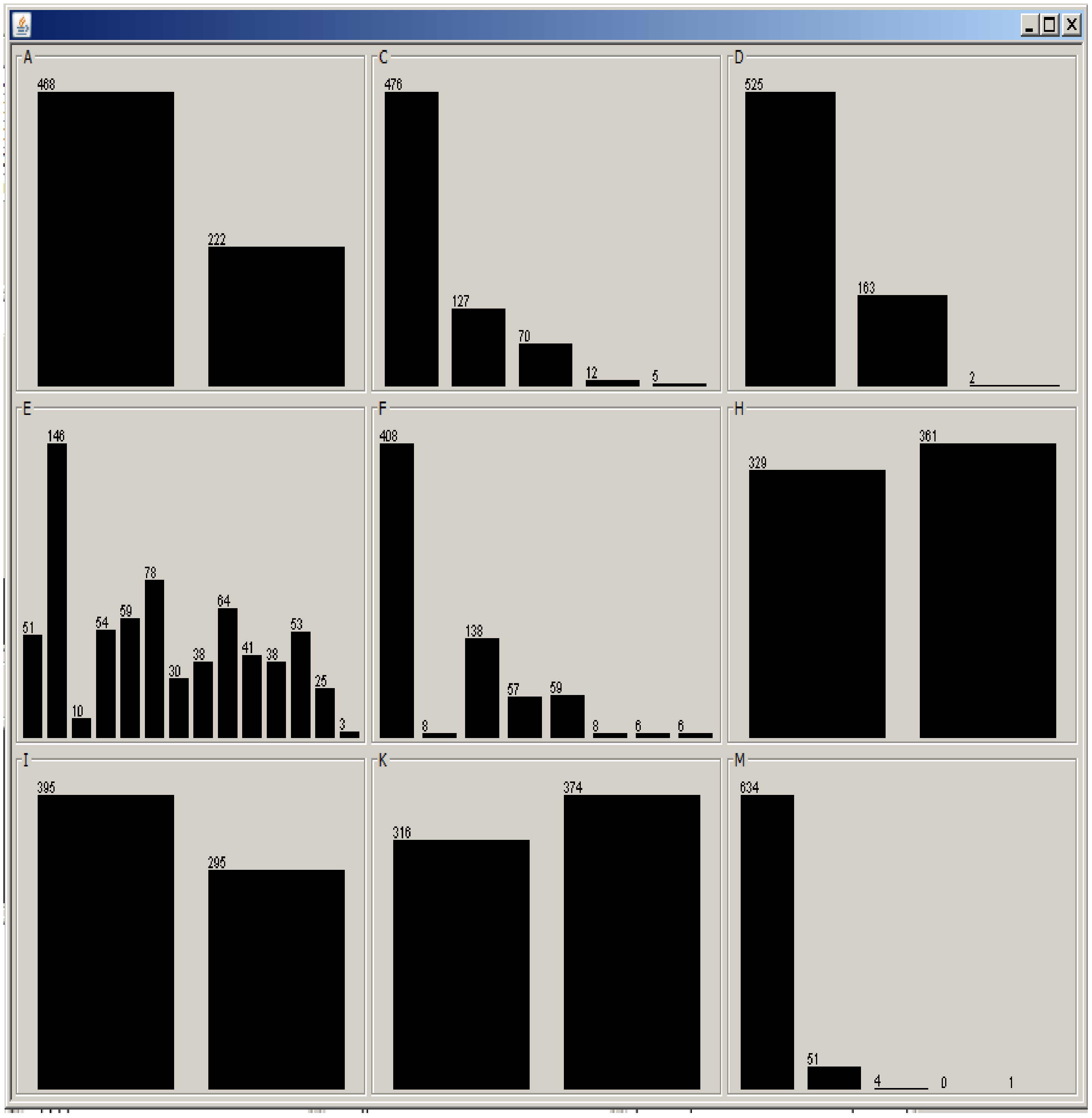
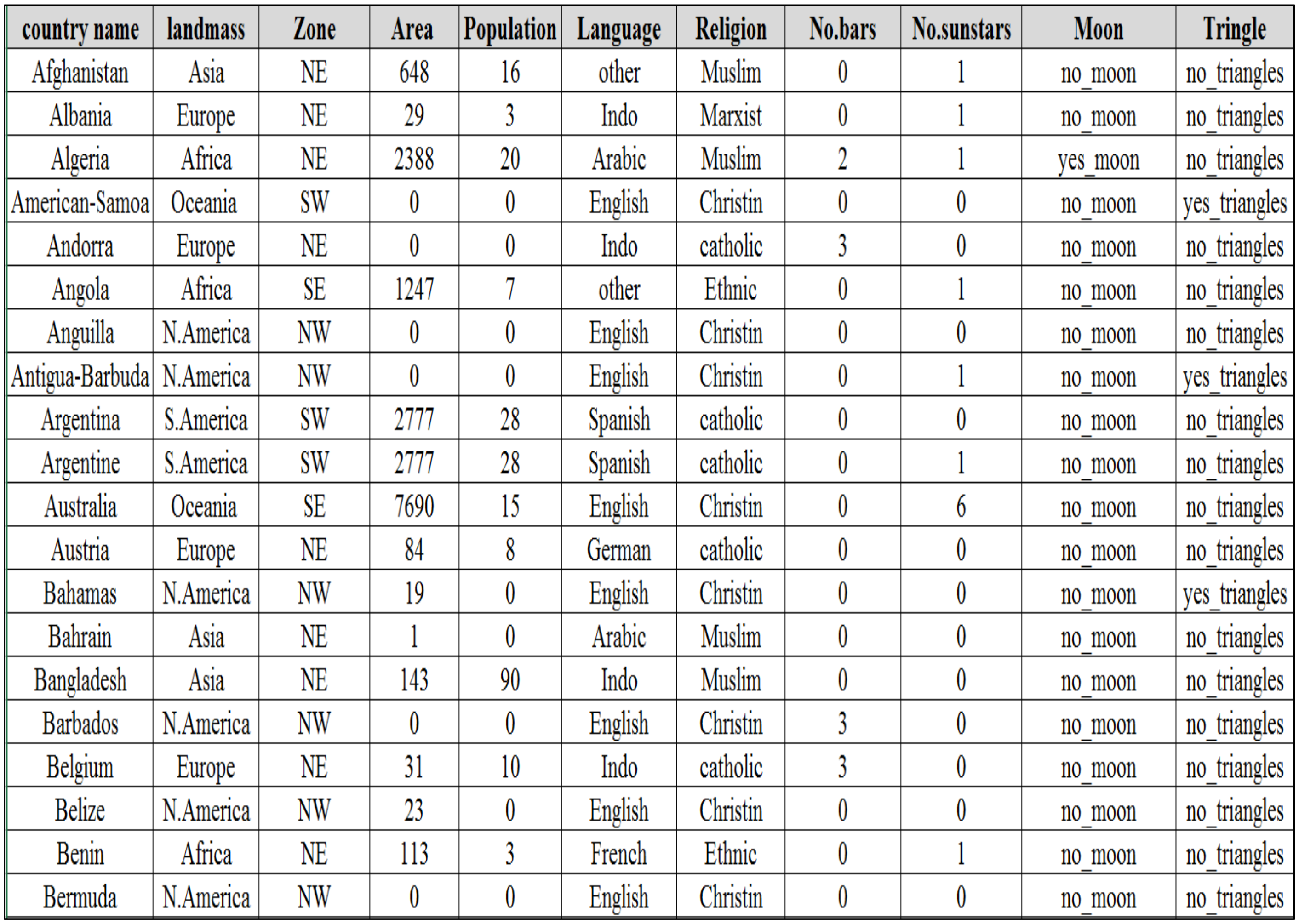


Figure 1: Data distribution of Categorical datasets *[10]*

Clustering is a subjective task, in which applying different algorithms on the same dataset may produce several different and meaningful clustering results. For example, Figure 2 presents some tuples in a mixed dataset, called flag. The dataset includes information about 194 different countries and their flags. Different clustering solutions can be obtained from this dataset based on different points of views (attributes). Different algorithms may group these countries into different meaningful clusters. For instance, an algorithm may identify a religious icon or symbols on the flag of the country and classify them. It is also possible to cluster these countries into not so useful groups based on the number of lines in the flags or presence of certain color in the flags.

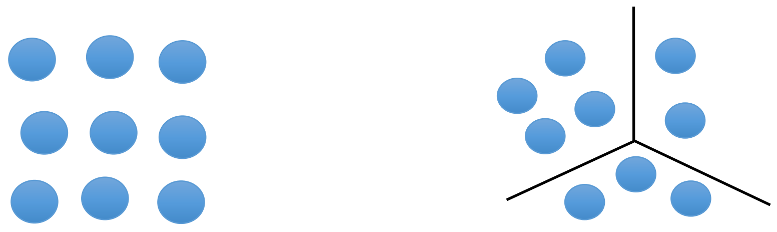


country name	landmass	Zone	Area	Population	Language	Religion	No.bars	No.sunstars	Moon	Tringle
Afghanistan	Asia	NE	648	16	other	Muslim	0	1	no_moon	no_triangles
Albania	Europe	NE	29	3	Indo	Marxist	0	1	no_moon	no_triangles
Algeria	Africa	NE	2388	20	Arabic	Muslim	2	1	yes_moon	no_triangles
American-Samoa	Oceania	SW	0	0	English	Christin	0	0	no_moon	yes_triangles
Andorra	Europe	NE	0	0	Indo	catholic	3	0	no_moon	no_triangles
Angola	Africa	SE	1247	7	other	Ethnic	0	1	no_moon	no_triangles
Anguilla	N.America	NW	0	0	English	Christin	0	0	no_moon	no_triangles
Antigua-Barbuda	N.America	NW	0	0	English	Christin	0	1	no_moon	yes_triangles
Argentina	S.America	SW	2777	28	Spanish	catholic	0	0	no_moon	no_triangles
Argentine	S.America	SW	2777	28	Spanish	catholic	0	1	no_moon	no_triangles
Australia	Oceania	SE	7690	15	English	Christin	0	6	no_moon	no_triangles
Austria	Europe	NE	84	8	German	catholic	0	0	no_moon	no_triangles
Bahamas	N.America	NW	19	0	English	Christin	0	0	no_moon	yes_triangles
Bahrain	Asia	NE	1	0	Arabic	Muslim	0	0	no_moon	no_triangles
Bangladesh	Asia	NE	143	90	Indo	Muslim	0	0	no_moon	no_triangles
Barbados	N.America	NW	0	0	English	Christin	3	0	no_moon	no_triangles
Belgium	Europe	NE	31	10	Indo	catholic	3	0	no_moon	no_triangles
Belize	N.America	NW	23	0	English	Christin	0	0	no_moon	no_triangles
Benin	Africa	NE	113	3	French	Ethnic	0	1	no_moon	no_triangles
Bermuda	N.America	NW	0	0	English	Christin	0	0	no_moon	no_triangles

.

Figure 2: A snapshot of the flag dataset [10]

As mentioned above, not every cluster is meaningful and useful. Poor clustering results are usually obtained because the clustering process is influenced by wrong or useless parameters. For example, Figure 3(a) shows nine objects, which look quite similar. Grouping these objects into one cluster seem to make more sense. However, if the clustering algorithm requires the user to provide the number of clusters as an input parameter, this may result in a poor cluster, as shown in Figure 3(b). Thus, the quality of clustering depends on the underlying similarity measure used in the clustering process and/or on the user-predefined parameters provided. Therefore, it is crucial (1) to find a good similarity/dissimilarity function, and (2) to identify influential and meaningful input parameters.



(a) Before clustering (b) After clustering

Figure 3: Example of (a) good clustering (b) not so useful clustering

1.2. Thesis Contributions

The contributions of this thesis are as follows:

1. Developing *k-mixed*, a technique for clustering mixed datasets. This technique modifies and uses an existing clustering algorithm, called *CLUC*, that works on categorical datasets.

2. Performing extensive experiments on different datasets to evaluate the performance of *k-mixed*.

3. Implementing the k-mean algorithm for mixed attribute types and comparing its performance with *k-mixed*. Results are also compared with those obtained using well-known algorithms

4. Evaluating the performance of the developed technique in various aspects, including detecting and handling noise, scalability, and sensitivity to the order of the data.

1.3. Thesis Outline

The rest of this thesis report is organized as follows. Chapter 2 presents the background concepts and techniques in cluster analysis. Chapter 3 is a review of the related work. In Chapter 4, we review the *CLUC* algorithm and discuss ways to improve its performance in terms of quality. Using the revised *CLUC*, Chapter 5 presents a design and implementation of *k-mixed*, a new algorithm proposed in this thesis for clustering mixed datasets. In Chapter 6, we report the results of our experiments for performance evaluation of *k-mixed* using real-life benchmark datasets. Chapter 7 includes concluding remarks and outlines the future work.

Chapter 2. Background, Definitions, and Concepts

2.1. Objects and Attributes

A dataset D is a collection of objects which consists of m columns and n rows. In clustering context, rows are the objects (also called tuples, records, observations, or data points) to be clustered into groups, and the columns are the attributes (also called domains, variables, or features). Each object X_i in D is an m-tuple, consisting of m attribute values. That is, $D = \{X_1, X_2, \dots, X_n\}$, where each X_i is a tuple of the form $(A_1, A_2, \dots, A_m)$, and each A_j is a value from the domain $DOM(A_j)$.

2.2. Clusters and Centers

Clustering or cluster analysis is a process of dividing a dataset D into a number of partitions or groups, called clusters. A cluster center may be a vector of values that represents the objects in the cluster. The center of a cluster is used to compare the similarity of a new object and decide whether it should be assigned to the cluster or not. It is important to note that the center could be an actual, real object in the dataset or a virtual object defined by, say, the mean of each attribute values in the cluster.

2.3. Similarity and Distance Measures

Similarities and distances functions/measures are opposite concepts used to describe quantitatively the closeness between two objects or two clusters. Generally, similarity functions are used when the attribute values are nominal or binary values. Normally, if the similarity of two objects is 1, it means that they are quite similar. On the other hand, if their similarity is 0, it means that they are dissimilar. For numeric data, we often apply distance functions. The distance between two similar objects is 0, and it is 1 if they are dissimilar. We will review major commonly used similarity and distance functions in Section 2.6.

2.4. Cluster Evaluation Methods

There are two methods to evaluate and assess the quality of clustering results [2]. These methods can be classified based on whether the ground truth is utilized[1] or not. If the ground truth is utilized, the external method is used, in which the clustering results are compared against the ground truth using a certain quality measurement. If the ground truth is not available, then the internal method is used.

- **External Methods**

Given clusterings C_1 and C_2, where C_1 is generated by an algorithm *G*, and C_2 is a portion that is provided from the ground truth, (*GT*, for short), there are two methods in general to evaluate and compare the clustering result of *G*: one based on counting pairs (e.g., Rand index, Jaccard index), and the other based on cluster matching (e.g., classification error) [11].

In the first methods to compare clusters C_1 and C_2, we need to define the following quantities:

- N_{11} Number of pairs of objects that are in the same cluster in both C_1 and C_2
- N_{01} Number of pairs of objects that are in the same cluster in C_2 but not in C_1
- N_{10} Number of pairs of objects that are in the same cluster in C_1 but not in C_1
- N_{00} Number of pairs of objects that are in different clusters in both C_2 and C_1

Based on the above four quantities, Rand index "*R*" and Jaccard index "*J* " are computed as follows:

$$R = \frac{N_{11} + N_{00}}{N_{11} + N_{01} + N_{10} + N_{00}}$$

$$J = \frac{N_{11}}{N_{01} + N_{10} + N_{11}}$$

[1] An ideal clustering often built or approved by a human expert.

In the second method, the term matching means counting the number of objects that grouped together under both *G* and *GT*. That is, the matching with the greater scores is selected and finally the scores of the matched clusters are aggregated to produce a total score.

- **Internal Methods**

In this approach, there is no a pre-specified structure used to evaluate the clustering results; only the structure and the features of the dataset are used. Several indices have been proposed to evaluate the clusters generated. These indices are applied based on the clustering method used during the clustering process (Hierarchical or partition-based). Examples of popular internal methods include, i.e., SSW (Sum of Squares Within the clusters), SSB (Sum of Squares Between the clusters), and Dunn's index. For details, interested readers are referred to [12].

Most of the work in the literature adopted the external method, in particular, they use the classification error measurement. In this work, we used the same for a fair comparison. We will explain and demonstrate this measurement method later in Chapter 6.

2.5. Classification of Data Types

Data types play a crucial role in clustering. In general, on the basis of their types, data can be classified into categorical data and numeric data.

2.5.1 Categorical Attributes

The domain *Dom*(A) of a categorical attribute A is a finite set of values, i.e., there is a one-to-one mapping from *D*(A) to {1,…,n}. Categorical data are also known as qualitative data. The values of categorical attributes are classified as follows [2]:

- **Nominal values**

In this type, the attribute values do not have a natural order; any order among nominal values is subjective and may have no importance. For instance, eye colors, phone numbers, zip codes, etc. A special type of nominal data is binary data, which has only two possible values. Binary data are divided into two categories: symmetric and asymmetric binary values. For symmetric data, both values are equally important; for example, the gender attribute of customers, which could take the male or female values. We may encode male to 1 and female to 0 or the other way around; both are equally fine. For asymmetric binary values, on the other hand, only the rare value is important which may be mapped to 1. Example of asymmetric binary value is the outcome of a disease test which could be positive or negative, one of which is crucial important and considered.

- **Ordinal Values**

Unlike nominal values, the order of ordinal values is important and meaningful. For instance, considering a set of ordinal values that represent the educational experience as elementary, high school, and college graduate. These values can be ordered from lowest level to highest level, with an order that is important and makes sense.

2.5.2 Continuous Types

Continuous data (also known as quantitative data) is an infinite set of values that can be ordered and measured. Continuous values can be classified into two categories: interval–scaled values and ratio scaled values. Interval–scaled values have no clear definition of the value zero and division cannot be employed on such values. For instance, the Graduate Record Examination (GRE) scores, temperature values, and calendar dates. On the other hand, ratio scaled values have absolute zero, for example, height, weight, and temperature in Kelvin.

2.6. Proximity Measures

Proximity measures are used to determine the closeness of two object O_i and O_j, or two clusters C_i and C_j, or between an object and a cluster. Depending on the data types, certain proximity measures may be applicable and used. In the following two sections, we review major commonly used measures for numeric and categorical data [2].

2.6.1 Measures for Numeric Data

- **Euclidean Distance**

The Euclidean distance is a widely used distance function, which computes the shortest path between a pair of two points. The distance between two points *x* and *y* in a d-dimensional space is defined as [2]:

$$dis\,(x,y) = \left[\sum_{j=1}^{d}(x_j - y_j)^2\right]^{\frac{1}{2}}$$

- **Manhattan Distance**

The Manhattan distance, also called *city block* distance, is the sum of the distances of all corresponding attributes. That is, the Manhattan distance between two points *x* and *y* in a d-dimensional is defined as [2]:

$$dis(x,y) = \sum_{j=1}^{d}\left|x_j - y_j\right|$$

- **Maximum Distance**

The maximum distance, also called the *sup* distance, is defined as the maximum value of the distances of the corresponding attributes (arguments). Formally [2],

$$dis(x,y) = \max_{1\le j\le d}\left|x_j - y_j\right|$$

- **Minkowski Distance**

The Minkowski distance is a generalization of Euclidean and Manhattan distances, and is defined as follows [2]:

$$dis(x, y) = \left[\sum_{j=1}^{m} \left|x_j - y_j\right|^r\right]^{1/r}$$

in which r is called the *order*. If r is set to 1, 2, and infinity, we get the Manhattan distance, Euclidean distance, and Maximum distance, respectively.

A distance measure is said to be metric if it satisfies the following properties [2]:

1. Non-negativity: *dis* (x, y)≥ 0
2. Reflexivity: $dis(x, y) = 0 \Leftrightarrow x = y$
3. Commutatively: $dis(x, y) = dis(y, x)$
4. Triangle inequality: $dis(x, y) \leq dis(x, z) + dis(z, y),$ for any point z.

2.6.2 Measures for Categorical Data

One way to define a similarity measure for categorical datasets is by defining an association among the values. For objects x and y in a dataset D of objects, different similarity functions can be defined using the association defined in Table 1.

	$a \in x$	$a \notin x$
$a \in y$	n_{11}	n_{01}
$a \notin y$	n_{10}	n_{00}

Table 1: An association table [8]

where n_{11} is the number of attribute values that are present in both *x* and *y*, n_{10} is the number of values that are in *x* but not in *y*, n_{01} is the number of values that are in *y* but not in *x*, and n_{00} is the number of values that are not present in either objects. If *x* and *y* are two objects, each being is a set of attributes values, then n_{11} $=|x \cap y|$,$n_{01} = |y - x|$, $n_{10} = |x - y|$, and $n_{00} = \big|V - |x \cup y|\big|$, where *V* is the union of all the attribute values in *D*.

The following similarity measures are commonly applied in text mining and information retrieval applications to find the distance between two sets of documents (or objects, in our context) [13], [8].

- **Simple Matching Coefficient**

This is useful when both positive and negative values used carry equal weights (i.e., symmetric values). For example, the values *male* and *female* for attribute gender.

$$sim(x,y) = \frac{(n_{11} + n_{00})}{(n_{11} + n_{01} + n_{10} + n_{00})} = \frac{|x \cap y| + \big|V - |x \cup y|\big|}{|V|} \quad (1)$$

- **Jaccard Coefficient**

Jaccard coefficient is a common similarity function used for binary variables, and is defined as the size of the intersection divided by the size of the union of the sample datasets. This function is usually used to detect plagiarism in documents.

$$sim(x,y) = \frac{n_{11}}{(n_{11} + n_{01} + n_{10})} = \frac{|x \cap y|}{|x \cup y|} \quad (2)$$

- **Dice Coefficient**

This measure is similar to Jaccard coefficient but assigns the matched values double the weight. It is commonly used to measure the similar of two strings on the basis of the number of common adjacent letters they include.

$$sim(x, y) = \frac{2 \times n_{11}}{n_{11} + n_{01} + n_{10}} = \frac{2 \times |x \cap y|}{|x \cup y|} \quad (3)$$

- **Overlap Coefficient**

This measure determines the similarity between two strings in terms of the number of shared words and the length of the strings.

$$sim(x, y) = \frac{|n_{11}|}{min(n_{01} + n_{10})} = \frac{|x \cap y|}{min(|x|, |y|)} \quad (4)$$

This work reverses and extends the CLUC algorithm [8], which designed its similarity based on the Dice Coefficient measure, defined in Equation 3, which is a well-known similarity measure [8]. It defines the similarity by the weighted intersection divided by the size of the union of the two sets. As will be shown in Chapter 4, CLUC extended the Dice coefficient to find the closeness between a bag and a collection of bags.

Any similarity function *sim* must satisfy the following three properties [2]:

1. $0 \leq sim\ (x, y) \leq 1$
2. $sim(x, x) = 1$
3. $sim(x, y) = sim(y, x)$

2.7. Classification of Clustering Methods

Given a set of objects to cluster, defining a “proper” similarity measure is crucial for the success of a clustering algorithm employed. There is no single measure for all types of datasets. Instead, a desired algorithm may be obtained by considering depending on the particular types of the input dataset and its attributes. In what follows, we review major clustering methods, [1].

2.7.1 Exclusive and Inclusive Clustering Methods

In *exclusive* method (also called *crisp* or *hard* clustering), each object belongs only to one cluster (see Figure 4), whereas in *inclusive* method (also called *fuzzy* or *soft* clustering) an object could belong to more than one cluster (see Figure 5). In the latter case, each object belongs to different clusters with different association degrees.

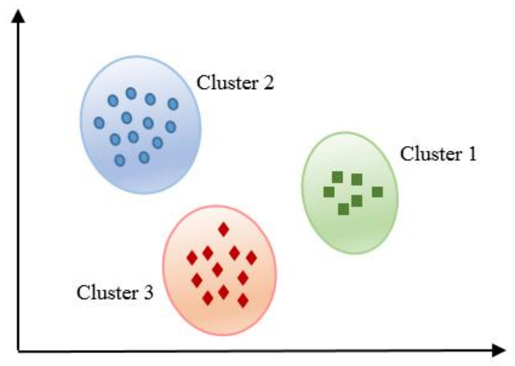


Figure 4: Hard/Crisp clustering

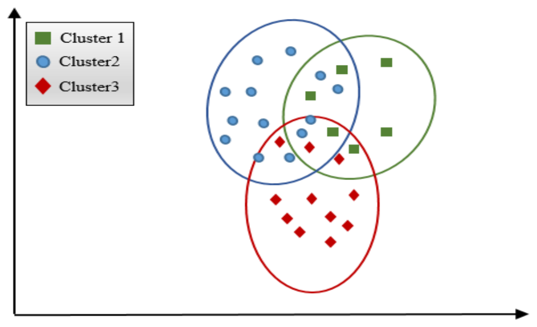


Figure 5: Soft/Fuzzy clustering

2.7.2 Hierarchical Clustering Methods

Hierarchical clustering is a sequence of data partitioning which aims to group objects into nested clusters. This method can be either agglomerative or divisive, depending on the strategy that is used for constructing the clusters (see Figure 6). In hierarchical agglomerative clustering method (HAC, for short), which is also referred to as bottom-up clustering method, clusters are formed by series of successive merge of clusters. The algorithm starts by allocating each object in a given dataset a cluster containing just that object. Then at each step of the merge performed in a bottom-up manner, the closest pairs of cluster are merged based on a distance or similarity measure considered. This process terminates when no more clusters could be merged.

In contrast to agglomerative method, a hierarchical divisive clustering method (HDC, for short) starts by a single cluster containing all the objects. Then at each step of the process, each cluster is further divided into smaller clusters. The process terminates when no more split is possible. Examples of well-known algorithms that adopt this method include *BRICH* [14], *ROCK* [7], and *CURE* [6].

As noted in [2] and [1], hierarchical top-down or bottom-up clustering algorithms suffer from the following two drawbacks.

1. No backtracking mechanism: Once a split or merge process is performed at each iteration step, there is no backtracking mechanism to correct possible poor quality clusters.

2. The memory utilization is high, making it not suitable for clustering high dimensional data.

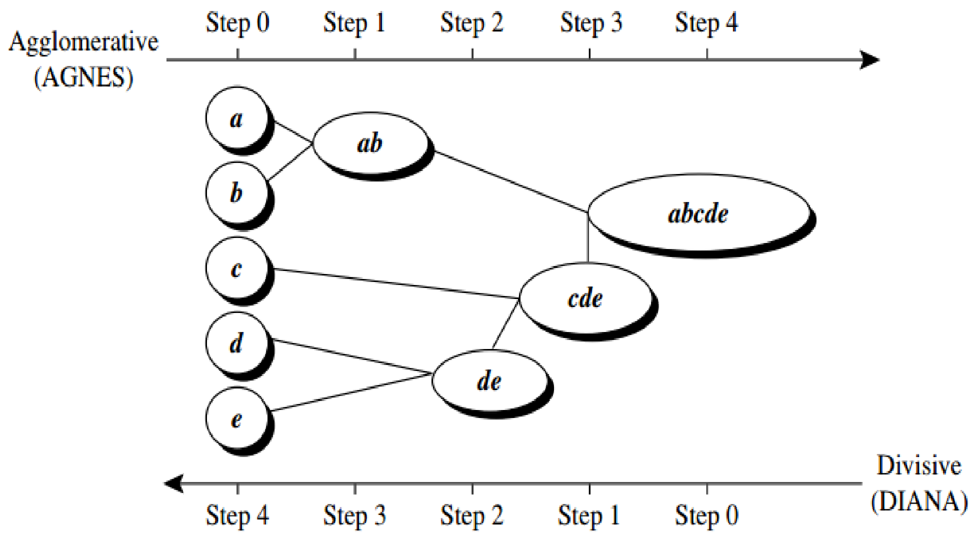


Figure 6: Example agglomerative and divisive hierarchical clustering on data objects {a, b, c, d, e} [1]

2.7.3 Partition-based Clustering Methods

In contrast to hierarchical clustering methods, a partitioning method is flat in creating one-level clusters. Partition-based algorithms are performed in two phases: initialization phase and portioning phase. In the initialization phase, *k* objects are randomly selected as centers for *k* clusters. Then the remaining objects are assigned to the closet center based on the similarity or dissimilarity function defined. Once an object is assigned to a certain cluster, the cluster center is updated. In the portioning phase, the dataset is scanned, objects are moved between the clusters, and the centers are updated. This phase is repeated until a termination condition is reached, which could be based on a limit imposed on the number of iterations or when there is no more changes in the clusters. *k-mean* [3] and *k-mode* [15] algorithms are well-known examples of partition-based clustering algorithms.

Although partition-based clustering is sufficient for clustering large datasets, it suffers from the following drawbacks and shortcomings [1]:

1. Its performance depends on the initialization phase.
2. Number of clusters k is a user defined parameter.
3. It is not effective for high dimensional data.
4. It is sensitive to noise and outliers.

2.7.4 Density-based Clustering Methods

In this method, clusters are defined as dense regions separated by regions of low density that considered as outliers or noise. The *DBSCAN* algorithm [4] is a typical example of density method, where data points in a given dataset are classified as either cores or borders, depending on the two user-predefined parameters *MinPts* and ε. While *MinPts* is the density threshold that specifies the minimum number of objects (the neighbors) within a cluster, ε specifies the radius of a neighborhood for every object. That is, a data point is considered as a core if it has more than *MinPts* neighbors within a radius ε; data points that are not cores are considered as borders. Density-based clustering algorithms are effective on high dimensional datasets, and can help discover arbitrarily shaped clusters.

2.7.5 Constraint-based Clustering Methods

In this method, the clustering process is guided by some user defined parameters, such as the desired number of clusters, the weight for different dimensions/attributes, the size of clusters, or other parameters that affects the clustering results. Constraints in cluster analysis can be divided into two categories [1]: constraints on objects and constraints on clusters. A constraint on objects defines how objects should be grouped into the same or different clusters.

On the other hand, a constraint on clusters defines some requirements on the clusters, such as the maximum or minimum number of objects in the clusters. In this thesis, we consider constraints on objects by taking into account the minimum similarity degree (cohesion) between objects within a cluster, provided as a user-defined parameter.

The main advantage of a constrained-based algorithm is that it contributes to generation of clusters which are expected to be more to user's satisfaction especially when a clustering algorithm is applied to high dimensional data [1].

2.8. Approaches to Clustering Data with Mixed Attribute Types

Existing algorithms for clustering data with mixed types of attributes can be classified in the following three approaches.

- **Conversion Approach**

In this approach, the input dataset of objects with mixed numeric and categorical attribute types is converted to a pure dataset in which the attributes are either numeric type or they are all categorical. We can then apply any desired clustering algorithm on the pure dataset obtained. There are two techniques for conversion: *encoding* and *discretization*. In encoding techniques, categorical values are mapped to numeric integer values. A dissimilarity function is then used to find the distance between object pairs. This works well if the categorical values have a natural order, for example the categorical attribute "size" with the values "small", "medium", and "large." On the other hand, encoding is a subjective task [11, 12, 13], and hence is not suitable for categorical attributes that have nominal values, such as the eye colors, which have no inherent order. Moreover, conversion of nominal data to numeric does not make sense and the proximity values are hard to interpret and justify.

Discretization is a technique to convert (or discretize) numeric values to nominal values which yields a pure dataset on which we can apply a desired categorical clustering algorithm. This technique is further classified into two categories on the basis of the availability of the class labels [16]. If labels are used in the clustering process, then the discretization technique is called *supervised*; otherwise it is called *unsupervised* discretization. Our focus in this research is mainly on the second strategy, unsupervised technique. In the literature of discretization, unsupervised methods are limited to equal width binning and equal frequency binning methods. In both methods, the set of values of a categorical attribute is partitioned into *k* number of bins (or intervals), for a pre-defined user parameter *k*. For the equal width method, the set of values is divided into *k* intervals with equal size, while for the frequency method; the set of values is divided into ranges having approximately the same number of continuous attribute values.

While some researchers expressed concerns about discretization for possible loss of important information, a number of sophisticated methods have been proposed to address the concerns. Examples of such methods include entropy-based discretization method (EBD) and Symbolic Nearest Mean Classifier (SNMC). For more details, interested readers are referred to [16].

- **Combined Approach**

In this approach, a single function is used to handle both numeric and categorical data types at the same time. The challenge here is to define a criterion function that can adequately capture the attributes that are more influential in deciding the clusters. This is done by assigning more weights to such attributes but this may cause biased results. The author in [17] believes that clustering using nominal values produces much better results than those produced by using mixed or pure numeric data.

- **Ensemble Approach**

Ensemble clustering combines the results of several runs of different clustering algorithms to produce the final clustering of the original dataset. This approach aims “for consolidating the results from a portfolio of individual clustering results” [18].

In the context of clustering data with mixed types of attributes, the ensemble approach is performed in three steps. In the first step, the input dataset is vertically divided into two parts, one for numeric attributes and the other for categorical attributes, both parts having the same number of records as the input dataset. Each of these two parts is then clustered separately using a suitable clustering algorithm for that type. The clustering results are then combined into a pure categorical or numeric data using a certain consolidating function based on consensus. Finally, a suitable algorithm is employed to cluster the combined dataset. Figure 7 summarizes the steps of this approach.

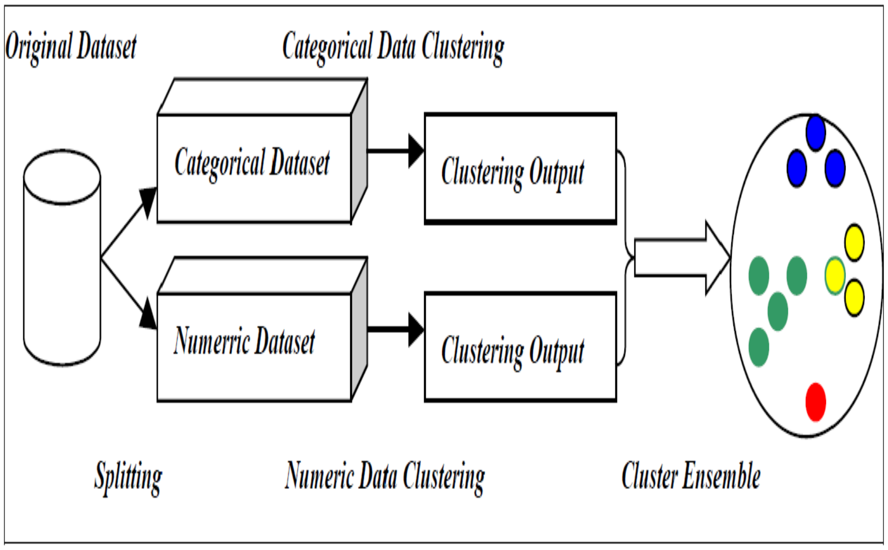


Figure 7: The ensemble process for mixed dataset [14]

2.9. Summary

In this chapter, we presented main definitions and concepts that will be used throughout this thesis. Then we reviewed some popular similarity / dissimilarity functions used for clustering and cluster analysis. We also looked at the classifications of the clustering methods and explained our decision for adopting a constraint-based clustering approach in our solution. We will next study existing approaches and solution techniques for clustering mixed datasets and in particular details on the discretization approach considered in this work.

Chapter 3. Related Work

As mentioned in Chapter 2, there are three main approaches for clustering data with mixed numeric and categorical attribute types: the combined approach, the conversion approach, and the ensemble approach. In this chapter, we review related work, focusing more on the first two approaches. For the following reasons, we did not consider the proposals that adopted the ensemble approach. First, the clustering methods used in some such proposals, like Fuzzy clustering method [14] were different from what we use in our work. Some algorithms are not comparable because they used synthetic datasets and the information of how the data was generated was not reported, i.e., [19].

3.1. Combined Approach Algorithms

3.1.1 K-prototype Algorithm

Huang proposed *K-prototype* algorithm [20] to cluster data with mixed attribute types. The proposed distance function defined below in Equation 5 is a combination of *k-mean* [3]and *k-mode* [15] algorithms. The formula defines the distance between a data object d_i and a cluster C_j. The squared Euclidean distance is used to find the distance between the numeric data. For categorical attributes, the simple match dissimilarity measure is applied, in which the distance between two values p and q is equal to zero if $p = q$, and equal to one, $otherwise$.

The distance between a data point d_i and a cluster C_j is defined as:

$$dis\left(d_i , C_j \right) = \sum_{t=1}^{m_n} (d_{it}^{n} - C_{jt}^{n})^2 + \gamma \sum_{t=1}^{m_c} \delta(d_{it}^{c} , C_{jt}^{c}) \qquad (5)$$

where m_n is the number of numeric attributes, and m_c is the number of categorical attributes. The mean vector $\mathrm{C}_{\mathrm{jt}}^{\mathrm{n}}$ is the center that represents the numeric attributes in cluster C_j, and the mode vector $\mathrm{C}_{\mathrm{jt}}^{\mathrm{c}}$ is the center for the categorical attributes in cluster C_j.

The overall distance measure denoted by this equation is the sum of the numeric distances and the weighted categorical distances.

K-prototype requires two user-predefined parameters; the number of clusters k and a threshold γ that specifies the weight of the data type. A large value for γ indicates that the clustering process is dominated by the categorical attributes, while a small value indicates that the clustering process is influenced more by the numeric attributes. An external method is used in this algorithm to measure the goodness; a good clustering result was obtained when the value γ was in the range 0.5 to 1.4.

K-prototype has some shortcomings and limitations as follows [21]:

The cost function is sensitive to the value γ; improper value will generate undesired clustering results. In addition, the center C_{jt}^{c} of the categorical attributes is represented by a vector of the most frequent value of each attribute. The problem with such representation is that rare values will not get a chance to appear in the center whereas in some applications such values might be more valuable than the frequent ones. On the other hand, Huang's cost function assumes that all numeric attributes contribute equally toward the clustering process, however, this does not reflect the real life satiations where numeric attributes might have different weights. In addition, the binary distance used over categorical values may lead to poor conclusions. For example, consider a categorical attribute that represents the eye colors. Based on Equation (1), the distance between the brown and hazel is one and that between brown and blue is one as well, however, the colors brown and hazel are closer to each other than the colors brown and blue.

3.1.2 K-mean algorithm for clustering numeric and categorical data

The K-mean algorithm for clustering mixed attribute types was proposed by Ahmad and Dey [21]. Generally the algorithm was designed to overcome the shortcomings with the *k-prototype* algorithm [20]. The modifications are as follows: (1) instated of applying the simple match dissimilarity function to categorical data, they proposed using a new dissimilarity function based on the value co-occurrences.

For this, the distance between two distinct values x and y of a categorical attribute A_t is calculated based on their co-occurrences with other categorical attribute values. (2) The weight for the numeric attributes is not a predefined parameter, but determined from the data. (3) The center of the categorical values C_{jt}^{c} is represented by listing all the categorical attributes values in C_j and not by the most frequent values.

The distance between a data point d_i and a cluster C_j is defined as:

$$dis(d_i, C_j) = \sum_{t=1}^{m_n} \left(w_t\left(d_{it}^{r} - C_{jt}^{r}\right)\right)^2 + \sum_{t=1}^{m_r} (\delta\left(d_{it}^{c}, C_{jt}^{c}\right)) \qquad (6)$$

where $\delta\left(d_{it}^{c}, C_{jt}^{c}\right)$ is the co-occurrence distance for categorical attributes and $w_t\left(d_{it}^{r} - C_{jt}^{r}\right)$ is the weighted squared Euclidean distance for numeric attributes.

Although this algorithm works well for data with mixed attributes, the complexity of the algorithm is high and computing the significance for categorical attributes is time consuming [22]. In addition, k-mean for mixed dataset inherits the problem of k-mean for numeric data in which the number of clusters must be determined in apriori and this becomes more difficult in clustering mixed datasets [23].

3.1.3 Similarity Based Agglomerative Clustering

Li and Biswas proposed *SBAC* (Similarity Based Agglomerative Clustering) algorithm [24], adopting the agglomerative method. They also defined a similarity measure based on the idea of Goodall's similarity measure [25], in which a greater weight is assigned to those pairs of objects which exhibit a greater match for attribute values that appear less frequently in the data. To illustrate the idea, consider a nominal attribute a and the two pairs of objects (i, j) and (l, m), where object i and j have an identical value for attribute a i.e., $(V_i)_a = (V_j)_a$, and l and m have an identical value for the same attribute a, $(V_l)_a = (V_m)_a$, but the two values are different, i.e., $(V_i)_a \neq (V_l)_a$. If the value $(V_i)_a$ appears more frequently in the data than the value$(V_l)_a$, then the pair of objects (l, m) is considered more similar than the pair (i, j), and hence the similarity assigned to (l, m) is not less than that assigned to(i, j).

For a numeric attribute p, the similarity between two pairs of objects (i, j) and (l, m) is computed based on two factors: the magnitude of the difference of the attribute values and the uniqueness of the attribute value pair. Intuitively, the smaller the difference between two values $\left((V_i)_p, (V_j)_p\right)$, the fewer pairs of values fall in the interval defined by $(V_i)_p$ and $(V_j)_p$. When two pairs of values have equal magnitude, then the uniqueness of the interval is defined by counting the frequency of occurrences of all values encompassed by the pair of values.

Once the similarity is computed for each attribute, two different statistical techniques are used to combine the individual probabilities results. For numeric attributes, Fisher's χ^2 transformation [26] was used, and for nominal attributes, Lancaster's mean value χ^2 transformation was used [27].

According to [21] and [28] the algorithm works efficiently on mixed attributes type, however, the complexity of this algorithm is quadratic in the number of objects in the dataset, which makes it not quite efficient for clustering very large datasets.

3.1.4 Usm-Squeezer Algorithm

He, Xu, and Deng in [28] extended the *Squeezer* algorithm [29] to work with mixed datasets. They proposed two algorithms: namely *usmSqueezer* (Unified Similarity Measure based Squeezer) that adopts the combined approach, and *dSqueezer* (discretizing before using Squeezer) which adopts the conversion approach. In what follows, we provide a brief introduction to the *Squeezer* algorithm, followed by a review of the two extended algorithms.

Squeezer is a clustering algorithm designed for datasets with categorical attribute types. It is a constraint clustering algorithm where the clustering results is influenced by a given threshold, denotes by *s*, which specifies the similarity degree between the objects within a cluster.

The similarity between a data object d_i and a cluster C_j is computed as:

$$sim(d_i, C_j) = \sum_{t=1}^{m} \left(\frac{Sup(a_t)}{|C_j|} \right) \qquad (7)$$

where m is number of categorical attribute, a_t is the value of the attribute t, and $Sup(a_t)$ is the support of a_t in the cluster C_j which is the number of data objects in C_j that have this attribute value. The algorithm is summarized in Figure 8.

Algorithm Squeezer (*D*, *s*)

Input: Data set *D* and threshold *s*

Begin

```
1.   While (D has unread tuple) {
2.       get Current tuple (D)
3.       if (tuple.tid==1)  /* read first tuple
4.           Create a new cluster C and add tuple.tid to it.
5.       else{
6.           for each existed cluster C
7.               compute similarity (sim_max) between C and tuple.tid
8.           if sim_max >= s
9.               add tuple.tid to cluster C
10.          Else
11.              create a new cluster C and add tuple.tid to it.
12.      }
13.  }
14.  End
```

Figure 8: Squeezer algorithm [29]

For the *usmSqueezer* algorithm, a unified similarity function was designed for handling numeric and categorical attributes at the same time in the framework of the *Squeezer* algorithm. In this algorithm, all numeric values are normalized first in the range [0, 1] to ensure that the numeric attributes with large values do not dominate the clustering process. Then Manhattan distance is used to find the closeness between numeric attribute values. On the other hand, the similarity between categorical attribute values are computed similarly to the way it is computed in *Squeezer* algorithm. Thus, the similarity between a data object d_i and a cluster C_j is defined as:

$$Sim(d_i, C_j) = \sum_{t=1}^{m_r} |a_{it} - c_{jt}| + \sum_{t=1}^{m_c} \left(\frac{Sup(a_t)}{|C_j|}\right) \qquad (8)$$

where c_{jt} is the center of the numeric attributes values in C_j.

Although this work extends a good clustering algorithm, its effectiveness has not been fully demonstrated [30].

3.2. Conversion Approach Algorithms

In this section, we review the techniques for clustering data with mixed attribute types that follow the conversion approach. They include the *d-Squeezer* and the *TMCM* algorithms.

3.2.1 The d-Squeezer Algorithm

The *d-Squeezer* algorithm adopted the discretization approach and utilized the CBA (Classification based on association) software to transform the mixed data to a pure categorical data [31]. Once the mixed dataset is transformed into a homogeneous type, then the *Squeezer* algorithm described above is applied.

The *d-Squeezer* algorithm uses a supervised discretization method in which the class label is utilized. This in return makes the algorithm restricted only to labeled datasets. During the clustering process, the two Squeezer algorithms scan the given dataset once.

This is efficient in terms of execution speed, however, some objects could be assigned to wrong clusters in the first scan and hence another scan is required to reallocate them correctly. This negatively impacts on the final clustering results and on the algorithms' robustness to the order of the data in the input dataset.

3.2.2 The TMCM Algorithm

Shih, Jheng, and Lai in [32] proposed the *TMCM* algorithm (Two-steps Method for Clustering Mixed Categorical and Numeric Data). This algorithm combines two algorithms to generate the final clustering results, the agglomerative hierarchical clustering and the k-mean algorithms. For handling the mixed attribute types, the encoding approach is used.

In this algorithm, a numeric value is assigned to the categorical attribute values according to their relationship among data items. Once the mixed dataset is converted into a pure numeric dataset, the *k-mean* [3] algorithm is applied. The *TMCM* algorithm is performed in three phases. In data preprocessing phase, numeric attributes are normalized and the similarity between categorical attribute values is computed. In the second phase, numeric values are assigned to categorical data. The last phase in which the clustering is performed, the *k-mean* algorithm is applied to pure numeric dataset. Figure 9 shows these phases.

The TMCM Algorithm

Phase 1: Data preprocessing.

1. Read input data, and normalize numeric attributes.
2. Find the categorical attribute *A* with the most number of distinct items to be the base attribute. The items in this base attribute are defined as the base items.
3. Count the frequency of co-occurrence between every categorical item and every base item, and store these information in a Matrix *M*.
4. Using the information in *M* to build the similarity between every categorical value and base item, and store these information in a matrix *N*.

Phase 2 : Assigning numeric values to categorical values

5. Find the numeric attribute that minimalize the within group variance to base attribute. Assign mean of the mapping value in this numeric attribute to every base item.
6. Applying the information of similarity that is stored in matrix *N* to find the numeric values for every categorical item.

Phase 3: Clustering

7. Apply HAC clustering algorithm to group data set into k clusters (in this work k is set to the 1/3 of the number of objects.)
8. Calculate the centroid of every formed cluster, and add every categorical item to be additional attributes of centroid. The value of a new attribute is the number that objects in this cluster contains this item.
9. Applying k-mean clustering algorithm again to group formed clusters in step 7-step 8 into desired cluster *k* groups.

Figure 9: The TMCM algorithm [32]

Chapter 4. Proposed Techniques

This chapter is divided into two sections. We begin by a brief introduction to the *CLUC* clustering algorithm, which was proposed for clustering pure categorical data, and improve its performance by changing its similarity function. We then used the modified *CLUC* to develop *k-mixed*, a new technique for clustering data with mixed numeric and categorical attribute types.

4.1. CLUC Algorithm

CLUC (a cohesion-based clustering) [8] is an algorithm designed for clustering pure categorical data. It takes a user-defined parameter $0 \leq \alpha \leq 1$ that indicates the least cohesion among the objects within a cluster.

Let *D* be a set of *n* objects, each of which is a tuple of *m* categorical attributes. That is, $D = \{ O_1, O_2, \dots, O_n\}$ and $O_i = \{ v_{i,1}, v_{i,2}, \dots, v_{i,m}\}$. The *CLUC* algorithm partitions the objects in *D* into k clusters, where k is not fixed priory. The cohesion of objects in each cluster C_j is not less than the user-defined parameter α.

Let c_j be the center of cluster C_j, Val_j be the total number of values of attributes in cluster C_j, and $V(\,px)$ be the number of different values for the attributes in C_j. Then the *center* c_j is represented by the px values in C_j with their corresponding frequencies. That is, the center is defined as:

$$c_j = \{(px, freqency) |\, px \in C_j\} \qquad (9)$$

Definition 1 (Cluster information): For each clusterC_j, the cluster information is a data structure which is defined as {(Cluster ID, {(Center, Objects ID)}. Cluster information is used to keep track of objects assigned to the clusters, and to record the similarities between objects and clusters. Details of tracking and recoding information and processes are provided in Chapter 6.

Based on the above, the cohesion between an object O_i and a cluster C_j is computed as follows:

$$cohesion\left(O_i, C_j\right) = \sum_{v_i \in px} fr(px) + 1/\left(Val_j + m\right) \qquad (10)$$

where $fr(px)$ is the number of occurrences of the value px in the center c_j of C_j.

In general, *CLUC* proceeds in two phases, the initialization phase and the refinement phase. In the initialization phase, tuples in the input dataset are read sequentially into the main memory. For each object O_i, its similarity-based cohesion is computed with respect to all the existing clusters, and assigned to the cluster to which it has the highest cohesion not less than the threshold value α. If no such a cluster is found, then O_i is included in new cluster C_j which is then created and its center c_j is determined.

In the second phase, the dataset is scanned again to find the best cluster for each object O_i such that O_i scores the highest cohesion not less than α . Each time an object assigned to or removed from C_j, c_j is updated. This process continues until there is no change in the clusters. Figure 10 presents the steps of the *CLUC* algorithm.

```
Algorithm 1: $CLUC(D, \alpha)$

Input : dataset D and the cohesion value $\alpha$
Output : clustering of D
/* phase 1 – initialization */
1.  Read the next object O from D
2.  while not end of D
3.      Find a cluster $C_i$ such that $Co(O, C_i) \geq \alpha$ and $Co(T, C_i) \geq Co(O, C_j)$ for all $j \neq i$
4.      If $C_i$ is found, then assign O to $C_i$
5.          else create a new cluster $C_n$, assign O to $C_n$
6.      read the next object O from D
7.  end while
/* phase 2 – Refinement */
8.  repeat
9.      Moved = false
10.     read the next object O from D
11. while not end of D
12.     find a cluster $C_i$ such that $Co(O, C_i) \geq \alpha$ and $Co(O, C_i) \geq Co(O, C_j)$ for all $j \neq i$
13.     C = cluster (O)
14.     if $C_i \neq$ cluster (T) then move O to $C_i$; move = true and delete C if it is empty
15.             else if no cluster was found
                then        create a new cluster $C_{k;}$
                    move O to $C_{k;}$
                    moved = true;

16.         read the next object O from D
17. end while
18. until moved = false
```

Figure 10: CLUC algorithm [8]

We illustrate how the algorithm works using the following two examples, which use the datasets shown in Tables 2 and 3.

Example 1

Table 2 represents a toy dataset which consists of 5 tuples represented by 4 attributes. Attributes *X*, *Y*, and *Z* represent the values from different domains and the fourth attribute represents the cluster ID to which the object belongs.

Table 2: A toy dataset

Tuples	***X***	***Y***	***Z***	$\boldsymbol{C_j}$
T_1	g	g	f	C_1
T_2	m	g	f	C_1
T_3	x	g	t	C_2
T_4	g	m	t	C_2
T_5	s	s	f	C_2

For this dataset, $C_1 = \{< g, g, f >, < m, g, f >\}$ and $C_2 = \{< x, g, t >, < g, m, t >, < s, s, f >\}$. Thus,

$c_1 = \{g: 3, f: 2, m: 1\}$ and $c_2 = \{x: 1, m: 1, g: 2, s: 2, t: 2, f: 1\}$ are the centers for C_1 and C_2, respectively. Now suppose that $T_6 =< g, m, x >$ is a new object to be assigned to a cluster, and the given cohesion threshold is 0.60. Thus,

$$cohesion\ (T_6, C_1) = \frac{(4\ + 2 + 0\ )}{9} = 0.66$$

and

$$cohesion\ (T_6, C_2) = \frac{(3\ + 2 + 2\ )}{12} = 0.58$$

Based on the calculations above, T_6 has a stronger cohesion with C_1 than with C_2, and hence assigned to C_1.

Intuitively, identical values in different domains could mean different things and assigned different weights. However, the *CLUC's* similarity measure did not distinguish such differences by collecting categorical values as bags irrespective of the attributes they are representing. For instance, in Table 2, the value g (or m) occurring under the two attributes *X* and *Y* are considered and treated as identical. This problem is more serious when these identical values appear very frequently and under many different attributes. Such values usually negatively impact the importance of less frequent values which could be more important and meaningful. Most categorical clustering algorithms, including *CLUC*, considers categorical data types (nominal and binary) as one type and given them all equal weights. For example, attribute *Z* in Table 2 has two values and could be binary attributes, however, this attribute was treated similarly to the nominal attributes *X* and *Y*. We note that similar treatment of such attributes can result in poor clustering outputs. To illustrate this problem, consider the following collection of tuples in Table 3 for which we use *CLUC* to cluster.

Example 2:

Table 3: An instance of heart disease dataset

Tuple ID	Gender	Pain Types	Smoker	Fever	Cough	X-Ray Result
T_1	male	normal	yes	no	yes	abnormal
T_2	female	flat	no	yes	yes	normal
T_3	male	sharp	yes	yes	no	indefinite
T_4	male	normal	no	no	no	normal
T_5	male	flat	yes	no	no	abnormal
T_6	female	normal	no	no	yes	normal

Given Table 3 which includes two nominal attributes (Pain Type and X-Ray Result), and four binary attributes (Gender, Smoker, Fever, and Cough). In what follows, we present the steps of the clustering of this data, considering 0.6 as the cohesion parameter threshold.

- **Initialization Phase:**

1. T_1 is read, and since there is no cluster yet, T_1 is assigned to a new cluster C_1 and its center c_1 is $\{ male: 1, normal: 1, yes: 2, no: 1, abnormal: 1\}$.
2. T_2 is read. Cohesion $(T_2 , C_1) = 0.66$, so T_2 is assigned to C_1 and c_1 is updated with T_2 as: $c_1 = \{male: 1, female: 1, flat : 1, normal: 2, yes: 4, no: 2, abnormal : 1\}$.
3. T_3 is read. Cohesion $(T_3 , C_1) = 0.91$, so T_3 is assigned to C_1 and c_1 is updated with T_3 as: $c_1 = \{male: 2, female: 1, flat: 1, sharp: 1, normal: 2, yes: 6, no: 3, abnormal: 1, indefinite : 1\}$.
4. T_4 is read. Cohesion$(T_4 , C_1) = 0.54$, so T_4 is added to a new cluster C_2 with its cluster center $c_2 = \{male: 1,\ normal: 2, no: 3\}$.
5. T_5 is read. Cohesion$(T_5 , C_1) = 0.79$. Cohesion$(T_5 , C_2) = 0.58$, so T_5 is assigned to C_1 and c_1 is updated with T_5

   as:$c_1 = \{male: 3, female: 1, flat: 2, sharp: 1, normal: 2, yes: 7, no: 5, abnormal: 2, idefinte: 1\}$
6. T_6 is read . Cohesion$(T_6 , C_1) = 0.70$. Cohesion$(\mathrm{T}_6 , C_2) = 0.75$, so T_6 is assigned to C_2 and c_2 is updated with T_6. $c_2 = \{male: 1, femle: 2, normal: 4, no: 5, yes: 1\}$.

To this point, two clusters were generated C_1 with four objects (T_1, T_2, T_3, T_5), and C_2 with two objects (T_4, T_6)

- **Refinement Phase**

In this phase, the dataset is scanned repeatedly until no change in clusters.

1. T_1 is read . Cohesion$(T_1 , C_1) = 0.79$. Cohesion$(T_1 , C_2) = 1$, so T_1 is moved to C_2. c_1 and c_2 are updated.

   $c_1 = \{male: 2, female: 1, flat : 2, sharp: 1, normal: 1, yes: 5, no: 4, abnormal : 1,\ indefinite : 1\}$.

   $c_2 = \{male: 2, femle: 1, normal: 5, no: 6, yes: 3, abnormal: 1\}$
2. T_2 is read. Cohesion$(T_2 , C_1) = 0.77$. Cohesion$(T_2 , C_2) = 0.79$, so T_2 is moved to C_2 and the centers are updated.

   $c_1 = \{male: 2, , flat : 1, sharp: 1, yes: 3, no: 3, abnormal : 1, indefinite: 1 \}$

   $c_2 = \{male: 2, femle: 2, flat: 1, normal: 6, no: 7, yes: 5, abnormal: 1\}$.

3. T_3 is read . Cohesion (T_3 , C_1) $= 0.83$. Cohesion(T_3 , C_2) $= 0.60$. T_3 stays in C_1 .

4. T_4 is read. Cohesion (T_4 , C_1) $= 0.50$. Cohesion(T_4 , C_2) $= 0.62$. T_4 stays in C_2 .

5. T_5 is read. Cohesion(T_5 , C_1) $= 0.83$. Cohesion (T_5 , C_2) $= 0.60$. T_5 stays in C_1.

6. T_6 is read. Cohesion(T_6 , C_1) $= 0.50$. Cohesion(T_6 , C_2) $= 0.83$. T_6 stays in C_2.

Since two objects were moved, the dataset is scanned again:

1. T_1 is read. Cohesion (T_1 , C_1) $= 0.77$. Cohesion(T_1 , C_2) $= 0.87$. T_1 stays in C_2 .
2. T_2 is read . Cohesion (T_2 , C_1) $= 0.61$. Cohesion(T_1 , C_2) $= 0.79$. T_2 stays in C_2 .
3. T_3 is read . Cohesion (T_3 , C_1) $= 0.83$. Cohesion(T_3 , C_2) $= 0.66$. T_3 stays in C_1 .
4. T_4 is read . Cohesion (T_4 , C_1) $= 0.55$. Cohesion(T_4 , C_2) $= 0.62$. T_4 stays in C_2.
5. T_5 is read . Cohesion (T_5 , C_1) $= 0.83$. Cohesion(T_5 , C_2) $= 0.73$. T_5 stays in C_1.
6. T_6 is read . Cohesion (T_6 , C_1) $= 0.50$. Cohesion(T_6 , C_2) $= 0.83$. T_6 stays in C_2

The clustering process is stopped here since no object moved between clusters. The final clustering result is shown in Table 4.

Table 4: CLUC clustering final result

Cluster 1		Cluster 2	
Tuple ID	**Values**	**Tuple ID**	**Values**
T_3	male ,sharp ,yes ,yes ,no ,indefinite	T_1	male, normal ,yes ,no ,yes ,abnormal
T_5	male ,flat, yes, no, no, abnormal	T_2	female ,flat ,no ,yes ,yes ,normal
		T_4	male ,normal ,no ,no ,no ,normal
		T_6	female ,normal ,no, no ,yes, normal

With $\propto = 0.60$ and three times of iterations, CLUC's similarity measure partitions the dataset into two clusters: Cluster 1 with two objects T_3 and T_5, and Cluster 2 with four objects T_1 , T_2 , T_4 and T_6. As pointed earlier, CLUC's similarity measure does not distinguish between the values in different domains which cause in increasing the weight of certain values and reducing the other. By looking at Table 4, it is clear that the clustering process was influencing by the two values "no" and "normal". All the objects with these values were allocated into Cluster 1, although some should not be. For instance, T_4 is represented the case of a healthy patient which should be placed in a different cluster. However, because the value "no" appears three times in T_4, it was assigned to cluster1. Similarly to T_3 , which represents the case of a patient with unusual test result that would be more reasonable to be isolated form other patients.

The problem of such measure arises when the given dataset is large and there are many asymmetric attributes. In many cases unimportant value such as "no" will get greater weight than "sharp" or "indefinite" that rarely appear. This affects negatively the quality and the interpretation of the clustering result.

4.2. Proposed Technique (*k-mixed*)

The main contribution of this thesis is the design of an effective algorithm, called *k-mixed* for clustering datasets with mixed attribute types. Given such a dataset, in the first step, called the *discretization*, the numeric values in the data are converted into discrete "categorical" values, using equal width method. In the second step, we use *CLUC+*, obtained by modifying the similarity measure of *CLUC*, a clustering method proposed for pure categorical data.

4.2.1 The *CLUC*+ Algorithm

As pointed out in Section 4.1, we noted that the performance of the *CLUC* algorithm is negatively affected by two issues: (1) identical values that appear under different attributes, and (2) binary attributes. To overcome the shortcomings, we propose *CLUC+*, which addresses the first issue by labeling the attribute values with their attribute name (or number). This allows distinguishing between identical values under different attributes.

For the second issue on binary attributes, we propose to assign a greater weight to important and uncommon values, done in a similar way we assign weights to nominal values. That is, for symmetric attributes, both values are taken into account during the clustering process. For asymmetric attributes, rare values are considered to be important and hence assigned greater weights. In the same context, frequent values get small weights that is equal to 1. This strategy is (1) to ensure that different types do not contribute equally towards the clustering process, and (2) to prevent unimportant but high frequent values from dominating the clustering result.

To illustrate the idea, consider again Example 1 in Section 4.1. If we know that a binary attribute Z is asymmetric, then we can redefine the centers of clusters C_1 and C_2 as follows:

$$c1 =< \{g1{:}\,1, g2{:}\,2, m1{:}\,1\}, \{f3\} >$$

and

$$c2 =< \{\, g1{:}\,1, g2{:}\,2, x1{:}\,1, s1{:}\,1, s2{:}\,1, m2{:}\,1, t3{:}\,2\}, \{f3\} >$$

Here we describe how *CLUC+* works. Consider a categorical dataset *D* which was divided by our proposed algorithm into *k* number of clusters. Each cluster C_j has a center denoted by c_j, which is an object represented by Val_j values of attributes in C_j. Suppose that *D* has *m* categorical attributes which consist of nominal attributes, symmetric binary attributes, and asymmetric binary attributes. For this, we divided the values of these attributes into two categories: important values and less important values. Let us refer to the number of values of the first category as (*ns*) which includes: nominal values, symmetric binary values, and the rare values in the asymmetric attributes. The second category includes the most frequent values in the asymmetric attributes which we denote by *(ab).* Now if $V(a_l)$ is the number of the different values of the l^{th}attribute in C_j, then the center c_j is formed by two parts. One part represents the important values (*ns*), formed by the $V(a_l)$ values for each attribute *l* with its frequency. The other part represents the less important values (*ab*), formed by listing their values. The cluster center c_j is then constructed as:

$$c_j = \{ < \{a_{l,ns} , freqency\}, \{a_{l,b}\} > \mid a_{l,m} \in C_j\} \qquad (11)$$

Using this, the cohesion between an objects O_i and a cluster C_j is then determined as:

$$cohesion\,(O_i, C_j) = \left(\sum_{l=1}^{ns} N_i + \sum_{l=1}^{ab} x_i \right) \Big/ \left(Val_j + m\right) \qquad (12)$$

where

$$N_i = \begin{cases} fr(a_l) + 1 & if\ \ (v_{i,l}) = (a_l)\ and\ (v_{i,l}) \in c_j \\ 0 & otherwise \end{cases}$$

and

$$x_i = \begin{cases} 1 & if \quad (v_{i,l}) = (a_l) \; and \; (v_{i,l}) \in c_j \\ 0 & otherwise \end{cases}$$

It is important to note that there are two cases where asymmetric attributes are treated as nominal:

1. If the dataset is a pure binary dataset, then there is no biased treatment by our definition.
2. If the two values have the same frequency and we do not know which one is more important.

Let us consider again Example 2 in Section 4.1, and assume that attribute "Smoker" and "Cough" satisfy case 2 above, where we do not know which of the two {yes, no} values for attribute "Smoker" is more important. Likewise, for "Cough", the number of occurrences of both values are the same. Then the clustering process proceeds as follows:

- **Initialization Phase**

1. The first item/object T_1 is read from the dataset and since no cluster has yet being created, , T_1 is assigned to a new cluster C_1 with its center c_1 as:

   $c_1 = \{male1{:}\,1, normal2{:}\,1, yes3{:}\,1, yes5{:}\,1, abnormal6{:}\,1\}, \{no4\}$.

2. T_2 is read, and its cohesion to C_1 is calculated to be 0.16 . The cohesion is less than the given threshold value which was 0.60 so a new cluster C_2 is created with a center

   $c_2 = \{female1{:}\,1, flat2{:}\,1, no3{:}\,1, yes4{:}\,1, yes5{:}\,1, normal6{:}\,1\}$

3. T_3 is read . Cohesion (T_3, C_1) = 0.33. Cohesion (T_3, C_2)=0.0 , so T_3 creates a new cluster C_3 with a center

   $c_3 = \{male1{:}\,1, sharp2{:}\,1, yes3{:}\,1, yes4{:}\,1, no5{:}\,1, indefinite6{:}\,1\}$

4. T_4 is read . Cohesion (T_4 , C_1)= 0.16. Cohesion (T_4 , C_2) =0.50 . Cohesion (T_4 , C_3)=0.33, so T_4 creates a new cluster C_4 with a center c_4 $\{male1{:}\,1, normal2{:}\,1, no3{:}\,1, no5{:}\,1, normal6{:}\,1\}, \{no4\}$

5. T_5 is read . Cohesion (T_5, C_1)= 0.66. Cohesion (T_5, C_2)=0.0 . Cohesion (T_5 , C_3)=0.33

Cohesion(T_5 , C_4)=0.50 ,so T_5 is assigned to C_1 and updates c_1 .

$c_3 = \{male1: 2, normal2: 1, flat2: 1, yes3: 2, yes5: 1, no5: 1, abnormal6: 2\}, \{no4\}$

6. T_6 is read . Cohesion (T_6, C_1)= 0.11. Cohesion (T_6, C_2)=083 . Cohesion (T_6 , C_3)=0.00

Cohesion(T_6 , C_4)=0.00 ,so T_6 is assigned to C_2 and its center c_2 is updated with the T_6 .

$c_2 = \{female1: 2, normal2: 1, flat2: 1, no3: 2, yes5: 2, normal6: 2\}\{yes4, no4\}$

- **Refinement Phase**

In this phase, the dataset is repeatedly scanned until no change in clusters

1. T_1 is read and its similarity is computed with each existing cluster:

Cohesion (T_1 , C_1) $=$ $2male1 + 1\ normal2 + 2yes3 + 1no4 + 2yes5 + 2abnormal\ 6/12$ = 0.83.

Cohesion (T_1 , C_2) $= 0\ male1 + 2\ normal2 + 0\ yes3 + 1\ no4 + 3\ yes5 + 0\ abnoraml6/18$=0.33

Cohesion (T_1 , C_3) $2\ male1 + 0\ normal2 + 2\ yes\ 3 + 0\ no4\ + 0\ yes5 + 0\ abnormal6/12 = 0.33$

Cohesion $(T_1, C_4) = 2\ male1 + 2\ normal2 + 0yes3 + 1no4 + 0yes5 + 0\ abnoraml6/12 = 0.41$

T_1 scores the highest cohesion with C_1 , so it stays in C_1

2. T_2 is read and its similarity is computed with all the existing clusters:

Cohesion $(T_2 , C_1) = 0\ female + 2\ flat2 + 0\ no3 + 0yes4 + 2\ yes5 + 0normal6/18 = 0.22.$

Cohesion$(T_2 , C_2) = 2\ female1 + 1\ flat2 + 2\ no3 + 1\ yes4 + 2\ yes5 + 2\ normal/12 = 0.83$.

Cohesion$(T_2 , C_3) = 0\ female1 + 0\ flat2 + 0no3 + 2yes4 + 0yes5 + 0normal6/12 = 0.16.$

Cohesion$T_2 , C_4) = 0\ female + 0\ flat + 2\ no\ 3 + 0\ yes4 + 0\ yes5 + 2\ normal6/12 = 0.33.$

T_2 scores the highest cohesion with C_2 , so it stays in C_2

3. T_3 is read and its similarity is computed with all the existing clusters:

Cohesion$(T_3 , C_1) = 3\ male + 2\ normal + 3\ yes3 + 0\ yes4 + 2\ no5 + 0\ indefinte/18 = 0.55.$

Cohesion $(T_3 , C_2) = 0\ male1 + 0\ sharp2 + 0\ yes3 + 1\ yes4 + 0\ no5 + 0\ indefinte/12 = 0.08$

Cohesion$(T_3, C_3) = 1$.

Cohesion $(T_3, C_4) = 2\,male1 + 0\,sharp2 + 0\,yes3 + 0\,yes4 + 2\,no5 + 0\,indefinite/12 = 0.33$

T_3 stays in C_3

4. T_4 is read its cohesion is computed with all existing clusters:

Cohesion$(T_4\,, C_1\,) = 3\,male1 + 2\,normal2 + 0\,no3 + 1\,no4 + 2\,no5 + 0\,normal6/18 = 0.44$.

Cohesion$(T_4\,, C_2) = 0\,male1 + 2\,normal2 + 3\,no3 + 1\,no4 + 0\,no5 + 3\,normal6/18 = 0.50$.

Cohesion $(T_4\,, C_3) = 2\,male + 0\,normal + 0no + 0\,no + 0\,no + 0\,normal\,/12 = 0.16$

Cohesion $(T_4\,, C_4) = 1$

T_4 stays in C_4

5. T_5 is read and its cohesion is computes with all the existing clusters:

Cohesion $(T_5, C_1) = 2\,male1 + 1\,flat2 + 2\,yes3 + 1\,no4 + 1\,no5 + 2\,abnormal\,/12 = 0.75$

Cohesion$(T_5\,, C_2\,) = 0\,male1 + 2\,flat2 + 0\,yes3 + 1\,no4 + 0\,no5 + 0\,normal/18 = 0.166$.

Cohesion $(T_5\,, C_3) = 2\,male1 + 0\,flat2 + 2\,yes3 + 0\,no4 + 2\,no5 + 0\,abnormal6/12 = 0.50$.

Cohesion $(T_5\,, C_4) = 2\,male1 + 0\,flat2 + 0\,yes3 + 1\,no4 + 2\,no5 + 0\,abnormal/12 = 0.41$.

T_5 stays in C_1

T_6 is read and its cohesion is computes with all the existing clusters:

Cohesion $(T_6\,, C_1\,) = 0\,female1 + 2\,normal2 + 0\,no3 + 1\,no4 + 2\,yes5 + 0\,normal6/18 = 0.27$.

Cohesion $(T_6\,, C_2\,) = 2\,female1 + 1\,normal2 + 2\,no3 + 1\,no4 + 2\,yes5 + 2\,normal6/12 = 0.83$.

Cohesion $(T_6\,, C_3\,) = 0\,female1 + 0\,normal2 + 0\,no3 + 0\,no4 + 0\,yes5 + 0\,normal6\,/12 = 0.0$

Cohesion $(T_6\,, C_4\,) = 0\,female1 + 2\,normal2 + 2\,no3 + 1\,no4 + 0\,yes5 + 2\,normal6/12 = 0.58$.

T_6 stays in C_2

Since no object moved between in the last iteration, the clustering process terminates. The final clustering result is shown in Table 5.

Table 5: CLUC + final clustering process

Cluster Number	Attributes values						Tuple ID
Cluster1	male1	normal2	yes3	no4	yes5	abnormal6	T_1
	male1	flat2	yes3	no4	no5	abnormal6	T_5
Cluster2	female1	flat2	no3	yes4	yes5	normal6	T_2
	female1	normal2	no3	no4	yes5	normal6	T_6
Cluster3	male1	sharp2	yes3	yes4	no5	idefinite6	T_3
Cluster4	male1	normal2	no3	no4	no5	normal6	T_4

Analyzing the clustering result, we note that *CLUC+* has divided the dataset into four interesting and meaningful clusters as shown in Table 5. Cluster 1 consists of two objects T_1 and T_5. This cluster represents the case where the patients are non-smoking males, with no fever, and the result of their X-Ray is abnormal. Cluster 2 contains two objects T_2 and T_6. This cluster represents female patients who are non-smokers, have cough, and their X-Ray result is normal. Clusters 3 and 4 include one object each, T_3 and T_4, respectively. As these objects are clearly distinct and different, they are correctly being placed in different clusters. Cluster 4 represents the case of a healthy male patient. Although T_4 has three values in common with Cluster 1 (male1, normal2, no4), and has four values (normal2, no3, no4, normal6) in common with Cluster 2, it was placed in a separate cluster. This is because our modified similarity measure considers only "important" values, which returned C_3 and C_4 as outliers which can be removed. This indicates that *CLUC+* can produce meaningful and natural clusters.

In our experiments, reported in the next chapter, we observed that compared to *CLUC*, the *CLUC+* clustering algorithm converges faster, return high quality clusters, and produces more meaningful and useful clusters than *CLUC*.

4.2.2 Discretization Process

We adopt the discretization method proposed as a pre-processing strategy for clustering data with mixed data types. The reasons for our choice of this strategy are that discrete features enjoy more effective representation, and they are more compact, accurate, and faster than continuous ones. Moreover, discretization helps reduce and simplify the range of the data, which results in better understanding, explaining, and using of the output [16].

In our work, we use the equal width binning method of discretization, in which the range of attributes is divided into k number of intervals or bins with equal size. The number of bins is chosen based on the density of the numeric attribute values. That is, we use wider (fewer) bins if the density is low and use narrower (more) bins otherwise. Equation 12 shows how to compute the width of the bins, equation 13 shows how the boundaries are defined.

$$W = \frac{max_{A_i} - min_{A_i}}{k} \qquad (13)$$

where $(max_{Ai} - min_{Ai})$ is the range of the numeric attribute A_i, and k is the pre-defined number of bins.

$$min_{A_i} + w, min_{A_i} + 2w, \dots, min_{A_i} + (k-1)w \qquad (14)$$

The following example illustrates the discretization process [33].

Example 3:

Suppose that we have a numeric attribute A_i with 9 values $\{28, 18, 16, 12, 16, 4, 24, 26, 0\}$, and suppose the number k of bin is set to 3. The discretization process is performed as follows.

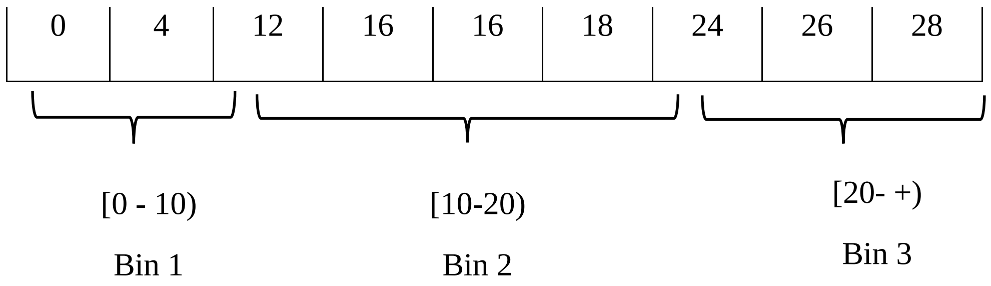
1.Numeric values are ordered ascending:

$$A_i =< \ 0, 4 ,12 ,16 ,16 ,18 ,24 ,26 ,28 >$$

2. The interval width is computed below using Equation 3:

$$w = \frac{28 - 0}{3} = 9.33$$

3. Numeric values are distributed over the bins based on the width:



4. Values within a bin are symbolized to a unique discrete value. This yields the final discrete values:

$$A_i = \{\mathrm{c, b, b, b, b, a, c , c , a}\}$$

4.3. Summary

In this chapter, we reviewed the *CLUC* algorithm and discussed its problems and limitations. We then introduced *CLUC+* which overcomes these shortcomings of *CLUC* and modifies its similarity measure. The major difference between the modified similarity function and *CLUC* lies on the fact that *CLUC+* handles nominal and binary data differently and gives them different weights, while *CLUC* doesn't distinguish between these two types of data. The results of our experiments indicate that *CLUC+* produces better clustering results. This motivates our use of *CLUC+* in the solution proposed in this thesis for clustering data with mixed attribute types. For this, we adopted the unsupervised discretization method with equal width binning method.

Chapter 5. Designing and Implementation of k-mixed

The *k-mixed* algorithm proceeds in three phases: data discretization, clustering, and evaluation, details of which are provided as follows.

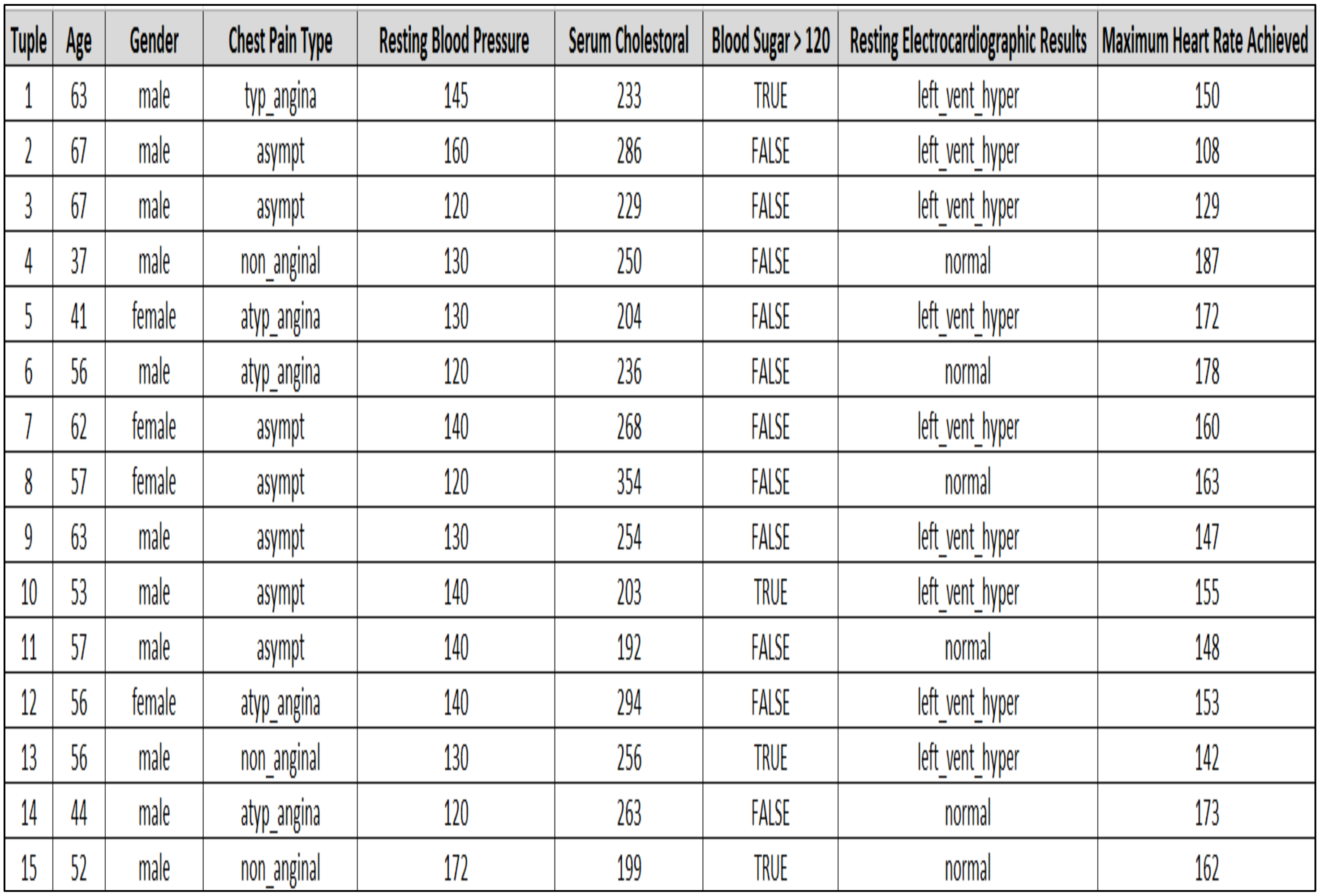
5.1. Discretization Phase

Discretization is a crucial step for the proposed algorithm, *k-mixed*. This work, will rely on Weka to convert the numeric attributes values into nominal values.

Weka (Waikato Environment for Knowledge Analysis) is a free analysis tool, which was built at the University of Waikato in New Zealand. It has a large collection of machine learning algorithms and visualization tools that ease the accomplishment of data mining tasks, i.e., data preprocessing, classification, clustering, regressions, features selection, etc. Weka saves the time of writing several lines of code and speeds up the discretization process. Figures 13 to 19 demonstrate the discretization steps. The user-friendly interface that Weka provides, helped us to understand the structure of the datasets and the ability to interpret the clustering results for different algorithms. We also utilized Weka in running a certain experiment to evaluate the robustness of our proposed algorithms (see Chapter 6).

Table 6 shows a sample of a real-life dataset [10]. As can be seen, there are 15 tuples, each of which is described by four numeric attributes (Age, Resting Blood Pressure, Serum Cholesterol, and Maximum Heart Rate) and four categorical attributes. More detail about the entire dataset is given in Chapter 6. The goal here is to illustrate how Weka can be used to discretize the numeric attributes and obtain a pure categorical dataset. To achieve this goal, the mixed dataset is uploaded into the Weka environment.

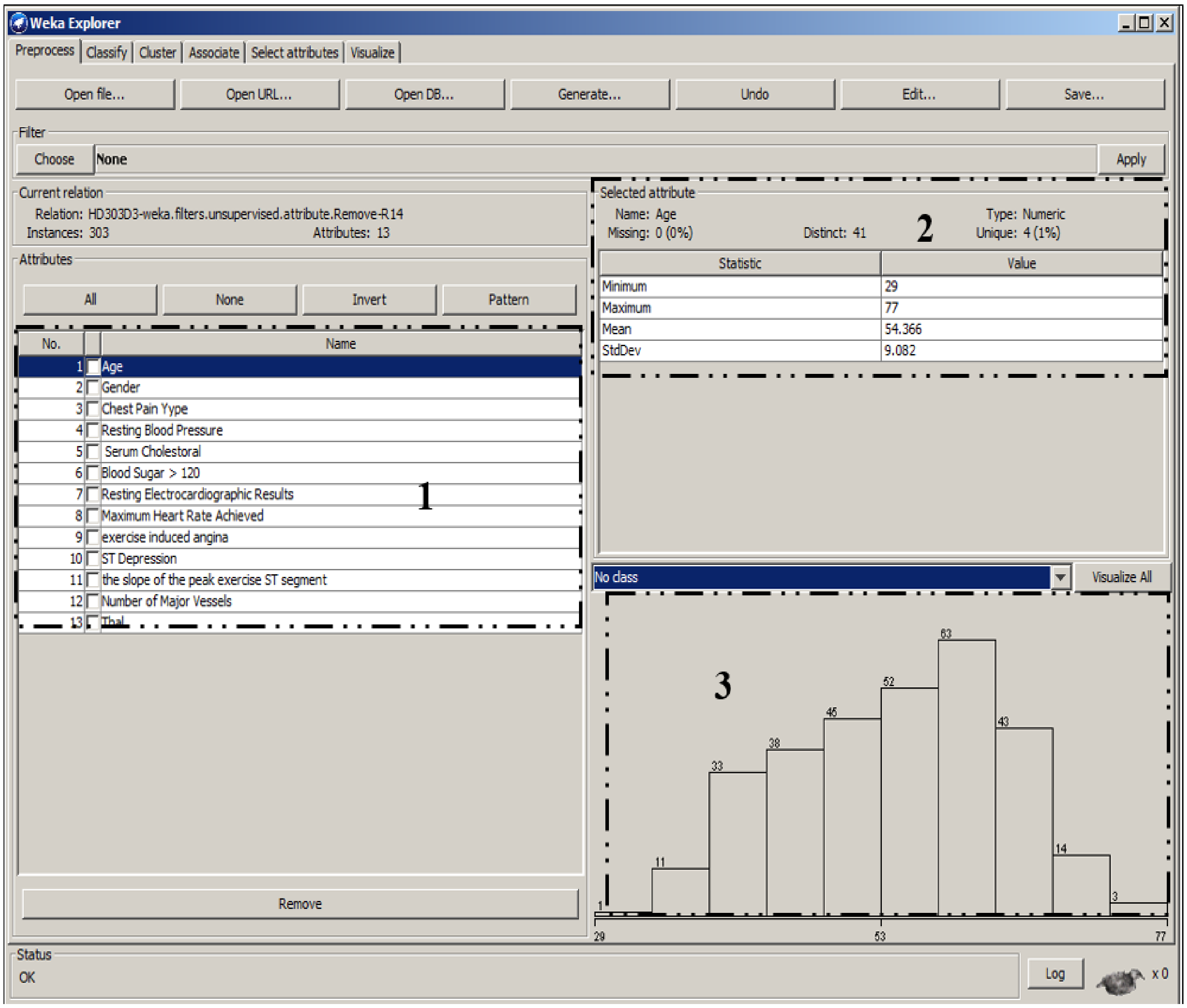
Table 6: Mixed Attributes Dataset



Tuple	Age	Gender	Chest Pain Type	Resting Blood Pressure	Serum Cholestoral	Blood Sugar > 120	Resting Electrocardiographic Results	Maximum Heart Rate Achieved
1	63	male	typ_angina	145	233	TRUE	left_vent_hyper	150
2	67	male	asympt	160	286	FALSE	left_vent_hyper	108
3	67	male	asympt	120	229	FALSE	left_vent_hyper	129
4	37	male	non_anginal	130	250	FALSE	normal	187
5	41	female	atyp_angina	130	204	FALSE	left_vent_hyper	172
6	56	male	atyp_angina	120	236	FALSE	normal	178
7	62	female	asympt	140	268	FALSE	left_vent_hyper	160
8	57	female	asympt	120	354	FALSE	normal	163
9	63	male	asympt	130	254	FALSE	left_vent_hyper	147
10	53	male	asympt	140	203	TRUE	left_vent_hyper	155
11	57	male	asympt	140	192	FALSE	normal	148
12	56	female	atyp_angina	140	294	FALSE	left_vent_hyper	153
13	56	male	non_anginal	130	256	TRUE	left_vent_hyper	142
14	44	male	atyp_angina	120	263	FALSE	normal	173
15	52	male	non_anginal	172	199	TRUE	normal	162

Figure 11 shows the main GUI (Graphic User Interface) of Weka. Once the dataset is uploaded, Weka will get insight into the dataset and display information about the data underlying structure. For example, the left panel, which is labeled with number one, lists the names of the attributes. Panel 2 on the right shows more detail about the selected attribute such as its type, number of distinct values, and the number of missing values. In addition, if the selected attribute is a numeric in type, then Weka computes and displays some basic statistical information, i.e. the mean and the standard deviation values. Moreover, panel 3 at the bottom right visualizes the distribution of the data of the selected attribute in a form of a histogram. The user can click on the button “visualize all” to see the distribution of all the attributes at once.

Having the mixed dataset uploaded, the numeric attributes are now ready to be discretized. By clicking on the button “Choose” right underneath the label “Filter” on the main GUI, an expanded windows will appear with a variety of options for operations, in which the “Discretize” option is selected (see Figure 12). Note that to this point the numeric data has not been converted yet.



Filter: None

Current relation
Relation: HD303D3-weka.filters.unsupervised.attribute.Remove-R14
Instances: 303 Attributes: 13

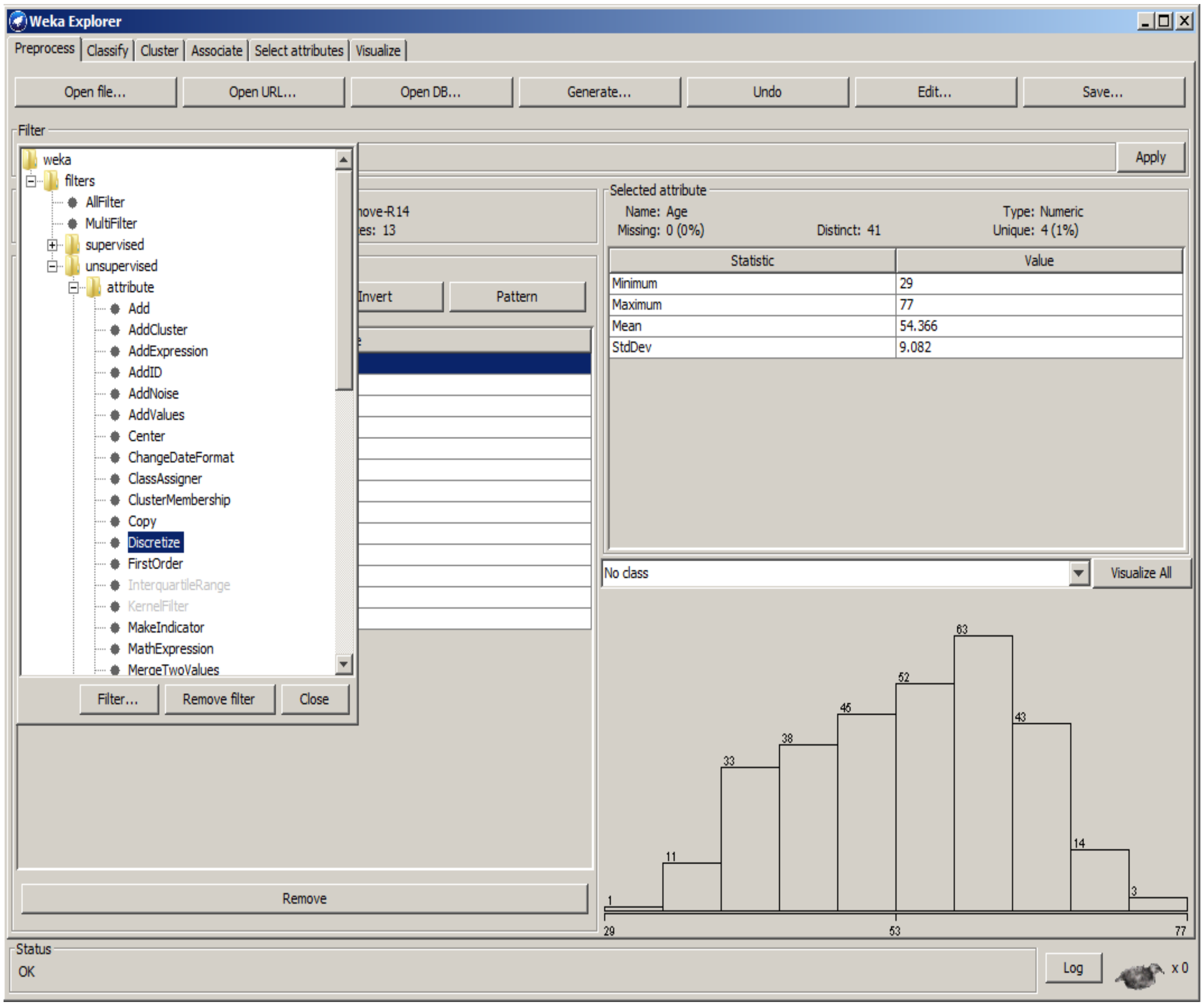
No.		Name
1	☐	Age
2	☐	Gender
3	☐	Chest Pain Type
4	☐	Resting Blood Pressure
5	☐	Serum Cholestoral
6	☐	Blood Sugar > 120
7	☐	Resting Electrocardiographic Results
8	☐	Maximum Heart Rate Achieved
9	☐	exercise induced angina
10	☐	ST Depression
11	☐	the slope of the peak exercise ST segment
12	☐	Number of Major Vessels
13	☐	Thal

Selected attribute
Name: Age Type: Numeric
Missing: 0 (0%) Distinct: 41 Unique: 4 (1%)

Statistic	Value
Minimum	29
Maximum	77
Mean	54.366
StdDev	9.082

Status
OK

Figure 11: Weka user interface



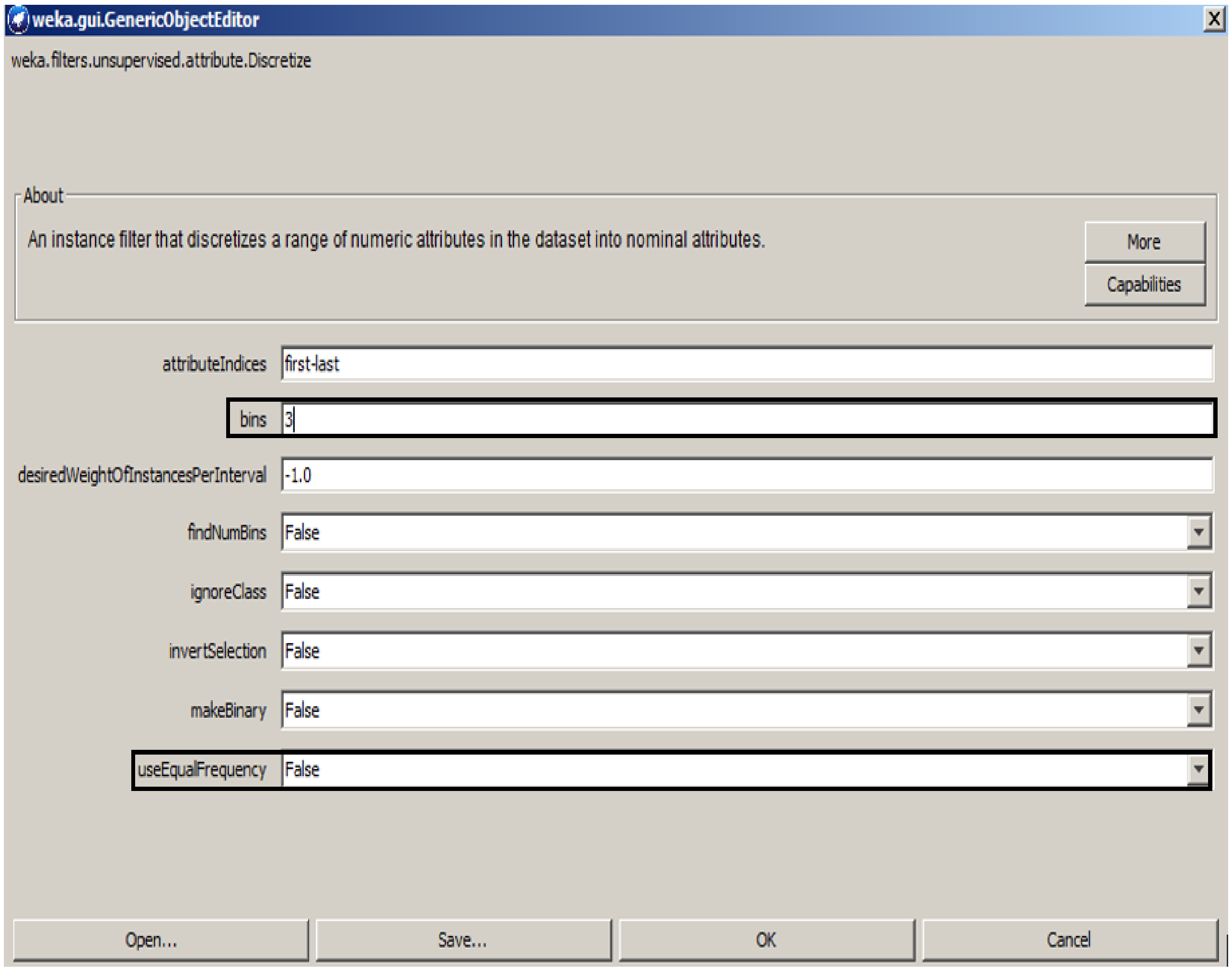
Selected attribute

Name: Age
Missing: 0 (0%)
Distinct: 41
Type: Numeric
Unique: 4 (1%)

Statistic	Value
Minimum	29
Maximum	77
Mean	54.366
StdDev	9.082

Figure 12: Select the “Discretize” option under the unsupervised category

As a result of selecting the “Discretize“ option in Figure 12, another windows will pop up to set the parammters for the discretization process (see Figure 13). As mentioned earlier, in this work we adopt the equal width method in which the number of intervals is a user input paramter, which in our implementation is determined in Weka through the parameter ”bins” and is applied to all numeric attributes at once. To specify the discretization method, the paramter ”useEqualFrequency“ is concidered. This paramenter has two values, “False” and “True”. If the paramenter is set to “False”, then the equal width binning method is applied, otherwise the equal frequency method is used. The rest of the paramters remin unchanged with their defult values as shown in Figure 13. The button “More” at the upper right corner of the GUI gives deatials about all the paramters (see Table 7).



weka.filters.unsupervised.attribute.Discretize

About

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

attributeIndices	first-last
bins	3
desiredWeightOfInstancesPerInterval	-1.0
findNumBins	False
ignoreClass	False
invertSelection	False
makeBinary	False
useEqualFrequency	False

Figure 13: Setting the parameters of the discretization process

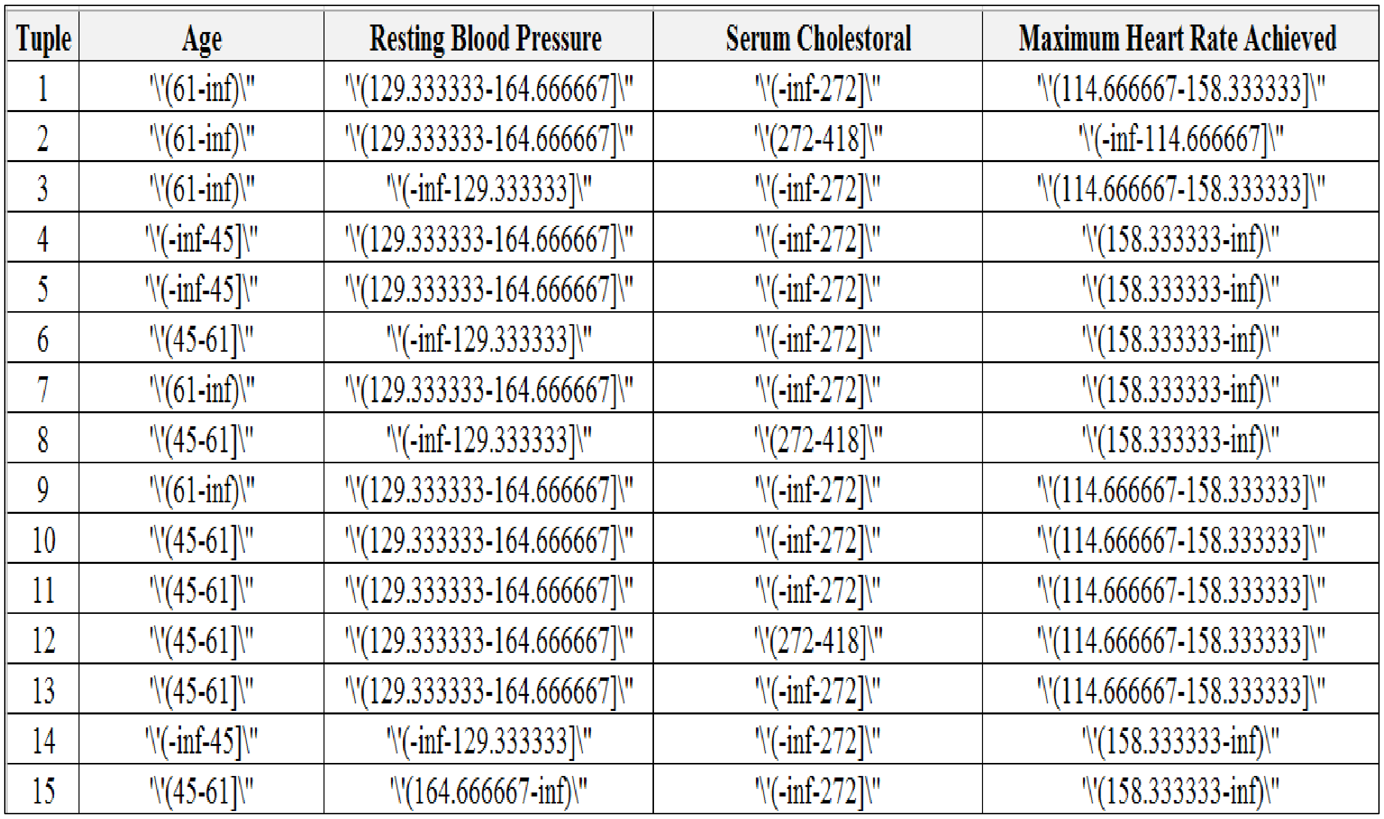
Table 7: A Description of the options avalible for “Discretize”

Option	Description
attributeIndices	Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".
bins	Number of bins.
desiredWeightOfInstancesPerInterval	Sets the desired weight of instances per interval for equal-frequency binning
findNumBins	Optimize number of equal-width bins using leave-one-out. Doesn't work for equal-frequency binning
ignoreClass	The class index will be unset temporarily before the filter is applied.
invertSelection	Set attribute selection mode. If false, only selected (numeric) attributes in the range will be discretized; if true, only non-selected attributes will be discretized.
makeBinary	Make resulting attributes binary.
useEqualFrequency	If set to true, equal-frequency binning will be used instead of equal-width binning.

Table 8 shows the result of the discretization process in Weka. To describe the result, consider the attribute “Age”. There are three intervals, the lower one is labeled as ‘\’(inf-45]\” which includes all values that are less than 45. The middle interval which is labeled as ‘\’(45-60]\” includes all the values from 45 and above but less than 61. The last interval is labeled as ‘\’(60-inf)\” which includes all the ages from 60 and above.

As can be seen in Table 8, the generated intervals are not in a proper format and since Weka does not have a feature to symbolize the generated intervals into unique discrete values. We had to do the symbolizing process manually in Microsoft Excel 2010.

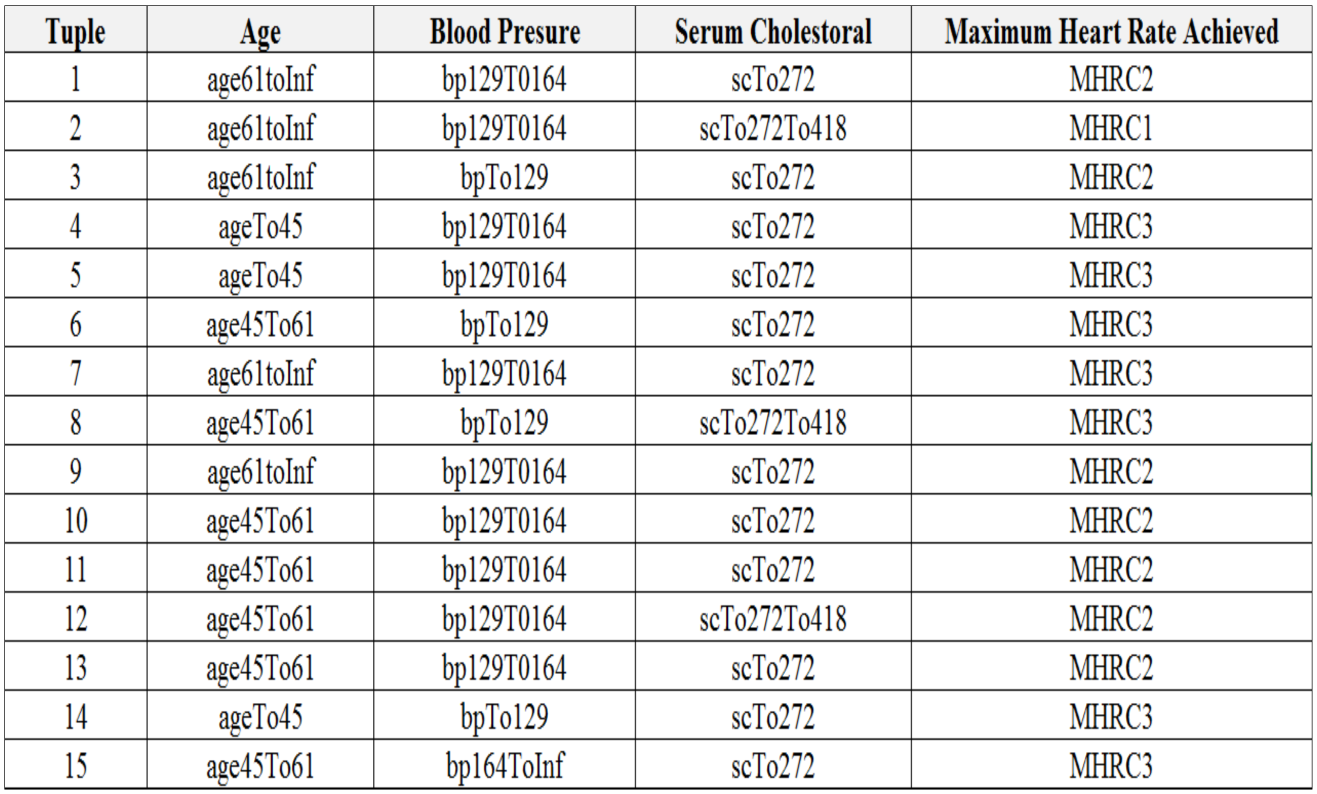
Table 8: Numeric Attributes after the discretization process



Tuple	Age	Resting Blood Pressure	Serum Cholestoral	Maximum Heart Rate Achieved
1	'\'(61-inf)\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
2	'\'(61-inf)\''	'\'(129.333333-164.666667]\''	'\'(272-418]\''	'\'(-inf-114.666667]\''
3	'\'(61-inf)\''	'\'(-inf-129.333333]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
4	'\'(-inf-45]\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''
5	'\'(-inf-45]\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''
6	'\'(45-61]\''	'\'(-inf-129.333333]\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''
7	'\'(61-inf)\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''
8	'\'(45-61]\''	'\'(-inf-129.333333]\''	'\'(272-418]\''	'\'(158.333333-inf)\''
9	'\'(61-inf)\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
10	'\'(45-61]\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
11	'\'(45-61]\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
12	'\'(45-61]\''	'\'(129.333333-164.666667]\''	'\'(272-418]\''	'\'(114.666667-158.333333]\''
13	'\'(45-61]\''	'\'(129.333333-164.666667]\''	'\'(-inf-272]\''	'\'(114.666667-158.333333]\''
14	'\'(-inf-45]\''	'\'(-inf-129.333333]\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''
15	'\'(45-61]\''	'\'(164.666667-inf)\''	'\'(-inf-272]\''	'\'(158.333333-inf)\''

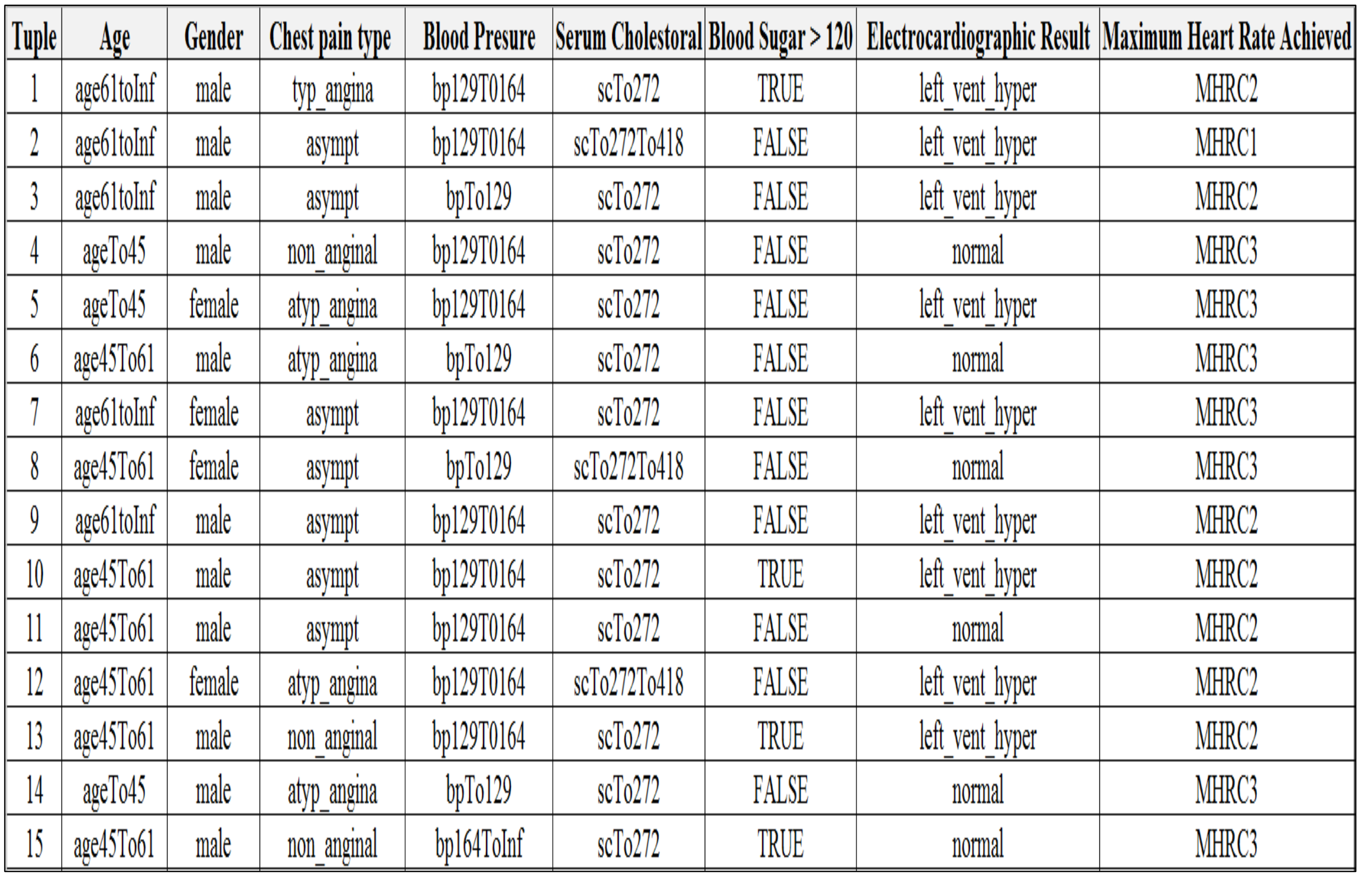
Table 9 shows the result of the symbolizing process in Excel. For this, we label each interval with a unique and meaningful discrete value. Basically, the new discrete values are a combination of the attribute's name and the interval's value. For example, consider the interval '\'(45-61]\" under the attribute "Age". In Excel all the values in this interval are symbolized to "age45To61". We do the same for the rest of the attributes. Note that the produced intervals by Weka could be used as they are; but we needed to customize it into a format that fits with our implementation. The final result is shown in Table 10.

Table 9: Symbolizing the interval values in Excel



Tuple	Age	Blood Presure	Serum Cholestoral	Maximum Heart Rate Achieved
1	age61toInf	bp129T0164	scTo272	MHRC2
2	age61toInf	bp129T0164	scTo272To418	MHRC1
3	age61toInf	bpTo129	scTo272	MHRC2
4	ageTo45	bp129T0164	scTo272	MHRC3
5	ageTo45	bp129T0164	scTo272	MHRC3
6	age45To61	bpTo129	scTo272	MHRC3
7	age61toInf	bp129T0164	scTo272	MHRC3
8	age45To61	bpTo129	scTo272To418	MHRC3
9	age61toInf	bp129T0164	scTo272	MHRC2
10	age45To61	bp129T0164	scTo272	MHRC2
11	age45To61	bp129T0164	scTo272	MHRC2
12	age45To61	bp129T0164	scTo272To418	MHRC2
13	age45To61	bp129T0164	scTo272	MHRC2
14	ageTo45	bpTo129	scTo272	MHRC3
15	age45To61	bp164ToInf	scTo272	MHRC3

Table 10: The final result, a pure categorical dataset



Tuple	Age	Gender	Chest pain type	Blood Presure	Serum Cholestoral	Blood Sugar > 120	Electrocardiographic Result	Maximum Heart Rate Achieved
1	age61toInf	male	typ_angina	bp129T0164	scTo272	TRUE	left_vent_hyper	MHRC2
2	age61toInf	male	asympt	bp129T0164	scTo272To418	FALSE	left_vent_hyper	MHRC1
3	age61toInf	male	asympt	bpTo129	scTo272	FALSE	left_vent_hyper	MHRC2
4	ageTo45	male	non_anginal	bp129T0164	scTo272	FALSE	normal	MHRC3
5	ageTo45	female	atyp_angina	bp129T0164	scTo272	FALSE	left_vent_hyper	MHRC3
6	age45To61	male	atyp_angina	bpTo129	scTo272	FALSE	normal	MHRC3
7	age61toInf	female	asympt	bp129T0164	scTo272	FALSE	left_vent_hyper	MHRC3
8	age45To61	female	asympt	bpTo129	scTo272To418	FALSE	normal	MHRC3
9	age61toInf	male	asympt	bp129T0164	scTo272	FALSE	left_vent_hyper	MHRC2
10	age45To61	male	asympt	bp129T0164	scTo272	TRUE	left_vent_hyper	MHRC2
11	age45To61	male	asympt	bp129T0164	scTo272	FALSE	normal	MHRC2
12	age45To61	female	atyp_angina	bp129T0164	scTo272To418	FALSE	left_vent_hyper	MHRC2
13	age45To61	male	non_anginal	bp129T0164	scTo272	TRUE	left_vent_hyper	MHRC2
14	ageTo45	male	atyp_angina	bpTo129	scTo272	FALSE	normal	MHRC3
15	age45To61	male	non_anginal	bp164ToInf	scTo272	TRUE	normal	MHRC3

5.2. Implementation of CLUC+

We developed *CLUC+* in the java programing language using a typical desktop computer with 6GB RAM running the Windows 7 OS. As explained earlier, *CLUC+* extends and improves *CLUC*, a pure categorical clustering technique CLUC, shown in Figure 14.

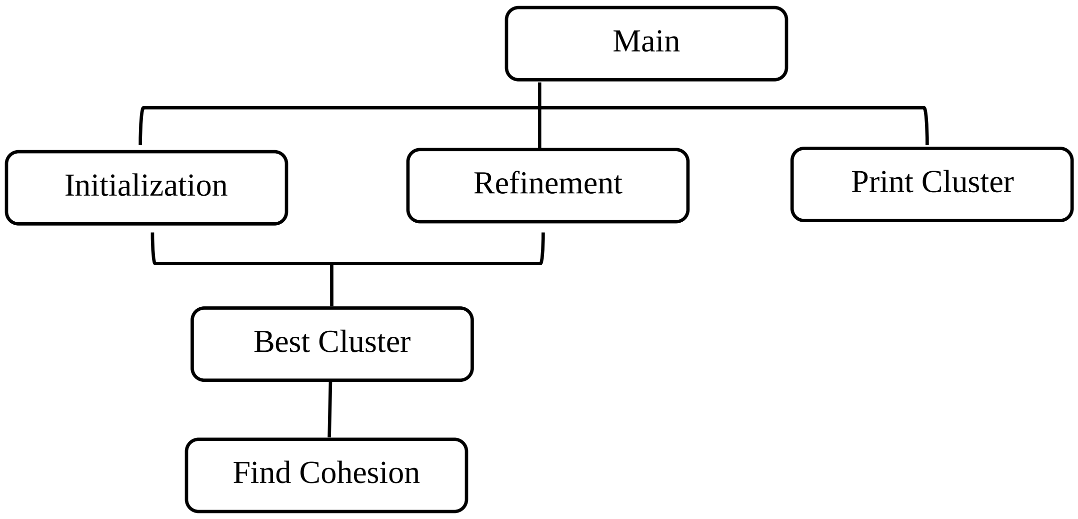


Figure 14: The CLUC+ design [8]

- **Main Module**

It is the main interface module between the user and the software clustering solution developed. It prompts the user to provide the name of the dataset and a cohesion threshold value. It uses the "Initialization" and the "Refinement" modules to cluster a given dataset. The "Print Cluster" module is used to display the clustering result.

- **Initialization Module**

This module scans the dataset once and produces the initial clusters based on the given cohesion value. This module utilizes the "Best Cluster" module in order to find the candidate clusters for objects.

- **Refinement Module**

In this module, the candidate clusters that generated from the initialization module are used as the input. The dataset is repeatedly scanned and the objects are reassigned to the best cluster to which it has the highest cohesion that is no less than the user-defined cohesion. Similarly to the “initialization“ module, this module uses the “Best Cluster“ module.

- **Print Cluster Module**

This module is responsible to display the final clustering result.

- **Best Cluster Module**

The best cluster module takes an object from the “Initialization “and the “Refinement” modules, and computes its cohesion with each existing cluster using “Find Cohesion“ module. Once the cluster is found or created, the cluster number is returned.

- **Find Cohesion Module**

It takes an object and the cluster ID from “Best Cluster” module, and computes the cohesion of the object to the cluster. It then returns the cohesion value.

5.3. Evaluation Phase

As mentioned in Chapter 2, we adopted an external method to evaluate the quality of the clustering results. For this, we used the R language **[34]** and utilized the package “Caret” that provides a number of useful functions and classes such as “confusionMatrix” and “table” functions which we used to assess the accuracy of our proposed solution.

Chapter 6. Experiments and Results

k-mixed was applied to different types of datasets taken from the benchmark data repository [10]. Note that all the benchmark datasets used in our work are classified and labeled by human experts. We compared the performance of our proposed techniques against existing clustering algorithms that use discretization as a pre-processing step and use the combined approach of clustering. The algorithms used in our comparison include: *k-prototype*, the two Squeezer algorithms (*d-Squeezer* and *usm-Squeezer*), and *k-mean for mixed numeric and categorical attributes type* (which we call as k-mean for mixed data, for ease of reference) algorithms. The comparison was conducted based on three aspects: accuracy, robustness, and efficiency (memory cost).

6.1. Evaluation of the Cluster Accuracy

For evaluation the accuracy of our proposed techniques, we used the external method. In particular we used the criterion based on cluster matching for which we find the clustering classification error. Example 4 shows how this value is computed.

Example 4

Suppose a given dataset D that consists of n objects and classified by human experts into k clusters, that is, $D = \{O_{1,j}, O_{2,j}, O_{3,j}, .., O_{n,j}\}$, where j denotes a cluster number to which an object belongs and j is one of the k values. Now suppose that an algorithm G is applied to D and generates p clusters, that is:$D' = \{O_{1,l}, O_{2,l}, O_{3,l}, .., O_{n,l}\}$, where$p = k$ $, and\ l \in p$. Thus, the clustering accuracy is defined as:

$$r = \sum_{i=1}^{k} a_i \Big/ n \qquad (15)$$

where n is the total number of objects in the given dataset D, and a_i is the number of objects in $\acute{D}$ that was correctly assigned to the corresponding clusters in D. Consequently, the classification error is computed as:

$$e = 1 - r \qquad (16)$$

That is, for two algorithms G_1 and G_2, the one that scores less classification error is considered to be better.

In this experiment, we run *k-mean for mixed data* and *k-mean* algorithms 100 times and the average is reported as the best clustering result. For *k-mixed*, in case of numeric and mixed datasets, we run the algorithm with different numbers of bins (#*B*) and these number vary from 3 to 10. For each run, $\propto$ is set to the value that produced the desired number of clusters and the best result generated from this is considered.

6.2. Robustness Evaluation

In the context of clustering, robustness refers to the stability of the algorithm performance to the order of the objects in the input dataset and to outliers and noise.

6.2.1 Insensitivity to input order

We evaluated the sensitivity of *k-mixed* to the sequence of the objects on two datasets with mixed attribute types. In our evaluation, we studied how the order of the objects could affect the number of generated clusters, and the quality of the clustering results. For this, the threshold value $\propto$ was set to the same value used in the experiments with the original order.

6.2.2 Detecting and Handling Outliers

In the context of clustering, outliers are objects that behave differently from the rest of the objects in the input dataset. Outlier detection is important in several applications including security, health care, and fraud detection. One of the methods to detect outlier objects is clustering-based method [1]. In this method, objects in small clusters or sparse clusters are defined as outliers and objects in large or dense clusters are considered as normal objects.

In our experiments, we used the Weka tool to identify the outlier objects in the dataset. Note that Weka uses the interquartile ranges method for filtering outliers.

6.3. Algorithm Efficiency

6.3.1 Handling Large Datasets

As with *CLUC*, a major advantage of *k-mixed* over *k-mean for mixed data* is its efficiency to handle datasets with large number of objects, because it doesn't require the entire dataset to be kept in the main memory but only the clusters information is maintained. For each cluster C_j we need to maintain the *IDs* of the objects that belong to that cluster, and also need a hash table to store the values with their corresponding frequencies. To this end, the memory required for k clusters and I attribute values of a given dataset D is $4I*k+4*|D|$. This method supports scalability of the proposed algorithm to large datasets easily, compared to using *k-mean for mixed data*. Besides, it allows effective clustering of datasets with large number of objects.

It is important to note that the performance of the *k-prototype* and the two Squeezer algorithms (*d-Squeezer* and *usm-Squeezer*) is not always compared with our technique, since the results were obtained from the respective publications and some of the aspects or the datasets were not demonstrated. For *k-mean for mixed data* algorithm, we implemented it and run it on our computer. For numeric datasets, we also compared our results against the simple k-mean, and for this we used the function k-means, which is implemented in the R language.

6.4. Benchmark Datasets

6.4.1 Categorical Datasets

- **Mushroom Dataset**

This dataset includes descriptions about 23 kinds of gilled mushrooms, introduced in Table 11. It includes 8124 records with 22 attributes. Each record is identified either as edible, or poisonous. This dataset contains 2480 missing values all for attribute number 11 and denoted by "?".

Table 11: Discerption of Mushroom dataset

Attribute Number	Attribute Name	Number of Distinct values	Attribute values
1	cap-shape	6	bell=b,conical=c,convex=x,flat=f,knobbed=k,sunken=s
2	cap-surface	4	fibrous=f,grooves=g,scaly=y,smooth=s
3	cap-color	10	brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4	bruises	2	bruises=t,no=f
5	odor	9	almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6	gill-attachment	4	attached=a,descending=d,free=f,notched=n
7	gill-spacing	3	close=c,crowded=w,distant=d
8	gill-size	2	broad=b,narrow=n
9	gill-color	12	black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
10	stalk-shape	2	enlarging=e,tapering=t
11	stalk-root	7	bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12	stalk-surface-	4	fibrous=f,scaly=y,silky=k,smooth=s

	above-ring		
13	stalk-surface-below-ring	4	fibrous=f,scaly=y,silky=k,smooth=s
14	stalk-color-above-ring	9	brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15	stalk-color-below-ring	9	brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16	veil-type	2	partial=p,universal=u
17	veil-color	4	brown=n,orange=o,white=w,yellow=y
18	ring-number	3	none=n,one=o,two=t
19	ring-type	8	cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20	spore-print-color	9	black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21	population	6	abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22	habitat	7	grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

- **Congressional Votes Dataset**

This dataset includes information about the United States Congressional Voting Records in 1984. The dataset consists of 435 records described by 16 attributes (see Table 12). This dataset is classified by human experts into two classes: Republicans and Democrat classes. Note that the attribute value "?" in this dataset doesn't mean a missing value; it means the member of parliament's decision is neither "yes" nor "no".

Table 12: Discerption of Congressional Vote dataset

Attribute Number	Attribute Name	Attribute values
1	handicapped-infants	y,n,?
2	water-project-cost-sharing	y, n,?
3	adoption-of-the-budget-resolution	y,n,?
4	physician-fee-freeze	y,n,?
5	el-salvador-aid	y,n,?
6	religious-groups-in-schools	y,n,?
7	anti-satellite-test-ban	y,n,?
8	aid-to-nicaraguan-contras	y,n,?
9	mx-missile	y,n,?
10	immigration	y,n,?
11	synfuels-corporation-cutback	y,n,?
12	education-spending	y,n,?
13	superfund-right-to-sue	y,n,?
14	Crime	y,n,?
15	duty-free-exports	y,n,?
16	export-administration-act-south-africa	y,n,?

- **Zoo Dataset**

This dataset consists of 101 tuples which describe 101 animals. Each attribute is described by 15 boolean attributes, one nominal attribute, and 2 numeric attributes. For the numeric attributes, one attribute represents the number of legs and the other one represents the "class label" (see Table 13). In this work, the attribute "leg" was treated as nominal since its values have a natural order.

Table 13 : Discerption of Zoo dataset

Attribute Number	Attribute Name	Attribute values
1	animal name	Unique for each instance
2	hair	Boolean
3	feathers	Boolean

4	eggs	Boolean
5	milk	Boolean
6	airborne	Boolean
7	aquatic	Boolean
8	Predator	Boolean
9	toothed	Boolean
10	backbone	Boolean
11	Breathes	Boolean
12	Venomous	Boolean
13	Fins	Boolean
14	Legs	set of values:{0,2,4,5,6,8}
15	Tail	Boolean
16	Domestic	Boolean
17	Catsize	Boolean

6.4.2 Numeric Datasets

- **Iris Dataset**

The discerption of this dataset is provided in Table 14. There are 150 records which classified by human experts into 3 classes (Virginica, Versicolour, and Setosa) each of which consists of 50 records.

Table 14 : Discerption of Iris dataset

Attributes Number	Attributes Name	Number of Distinct Values	Attribute Range
1	sepal length	35	[4.3-7.9]
2	sepal width	23	[2 -4.4]
3	petal length	43	[1-6.9]
4	petal width	22	[0.1-2.5]

- **Breast Cancer Dataset**

This dataset contains information about 699 records, describes by 10 attributes, introduced in Table 15. The data is classified into the “benign” class with 458 records, and “malignant” class with 457 records. Note that the first attributes in this dataset presents irrelevant values (patient ID number), and hence removed in our experiments

Table 15: Discerption of Breast Cancer dataset

Attributes Number	Attribute Name	Number of Distinct Values	Attribute Range
1	Clump Thickness	10	[1-10]
2	Uniformity of Cell Size	10	[1-10]
3	Uniformity of Cell Shape	10	[1-10]
4	Marginal Adhesion	10	[1-10]
5	Single Epithelial Cell Size	10	[1-10]
6	Bare Nuclei	10	[1-10]
7	Bland Chromatin	10	[1-10]
8	Normal Nucleoli	10	[1-10]
9	Mitoses	10	[1-10]

- **Wine Dataset**

Wine dataset consists of 178 records, each of which is described by 13 attributes (see Table 16). The dataset is classified into three classes numbered from 1 to 3.

Table 16 : Discerption of Wine dataset

Attributes Number	Attribute Name	Number of Distinct Values	Attribute Range
1	alcohol	126	[11.03-14.83]
2	malic acid	133	[0.74-5.8]
3	ash	79	[1.36-3.23]
4	alcalinity of ash	63	[10.6-30]
5	magnesium	53	[70-162]
6	total phenols	97	[0.98-3.88]
7	flavanoids	132	[0.34-5.08]
8	nonflavanoid phenols	39	[0.13-0.66]
9	proanthocyanins	101	[0.41-3.58]
10	color intensity	132	[1.28-13]
11	hue	78	[0.48-1.71]
12	OD280/OD315 of diluted wines	122	[1.27- 4]
13	proline	121	[278-1680]

6.4.3 Mixed Attributes Type Datasets

For mixed attributes type, we used credit approval dataset and two different heart disease datasets. We considered these datasets because they are widely used in related work and have a good mixture of attributes types.

- **Heart disease Datasets**

There are two different heart disease datasets, Cleveland and Statlog heart disease datasets. Both datasets are classified by human experts into two classes that represented the presence or absence of the disease.

For Cleveland dataset, there are 303 records each of which is described by 6 numeric and 7 categorical attributes. Of all the records in this dataset, 139 records are classified as “positive”, and 164 records are classified as “negative.” Table 17 presents the characteristics of this dataset. For the Statlog heart disease dataset, presented in Table 18, there are 270 records in total, of which 120 are identified and placed in the “presence” class, and the rest (150 records) are placed in the “absence” class.

Although the two datasets have the same set of attribute names and values, they differ in data distribution (see Figures 15 and 16).

Table 17: Description of Cleveland heart dieses dataset

Attribute Number	Attribute Type	Attribute Name	Attribute Values / Range
1	continues	age	[29-77]
2	binary	sex	Male-female
3	nominal	chest pain type	typical angina, atypical angina non-angina pain ,asymptomatic
4	continues	resting blood pressure	[94-200]
5	continues	serum cholesterol	[126-564]
6	binary	fasting blood sugar > 120 mg/dl	1= true, 0=no
7	nominal	resting electrocardiographic results	normal ,abnormal ,definite left ventricular hypertrophy
8	continues	maximum heart rate achieved	[71-202]
9	binary	exercise induced angina	1=true,0=no
10	continues	ST depression	[0-6.2]
11	nominal	the slope of the peak exercise ST segment	up-sloping ,flat ,down-sloping
12	continue	number of major vessels	[0-3]
13	nominal	thal	normal, fixed defect ,reversible defect

Table 18 : Description of Statlog heart dieses dataset

Attribute Number	Attribute Type	Attribute Name	Attribute Values / Range
1	continues	age	[29-77]
2	Binary	sex	0,1
3	nominal	chest pain type	1,2,3,4
4	continues	resting blood pressure	[94-200]
5	continues	serum cholesterol	[126-564]
6	binary	fasting blood sugar > 120 mg/dl	1= true, 0=no
7	nominal	resting electrocardiographic results	0,1,2
8	continues	maximum heart rate achieved	[71-202]
9	binary	exercise induced angina	1=true,0=no
10	continues	ST depression	[0-6.2]
11	nominal	the slope of the peak exercise ST segment	1,2,3
12	continue	number of major vessels	[0-3]
13	nominal	thal	3,6,7

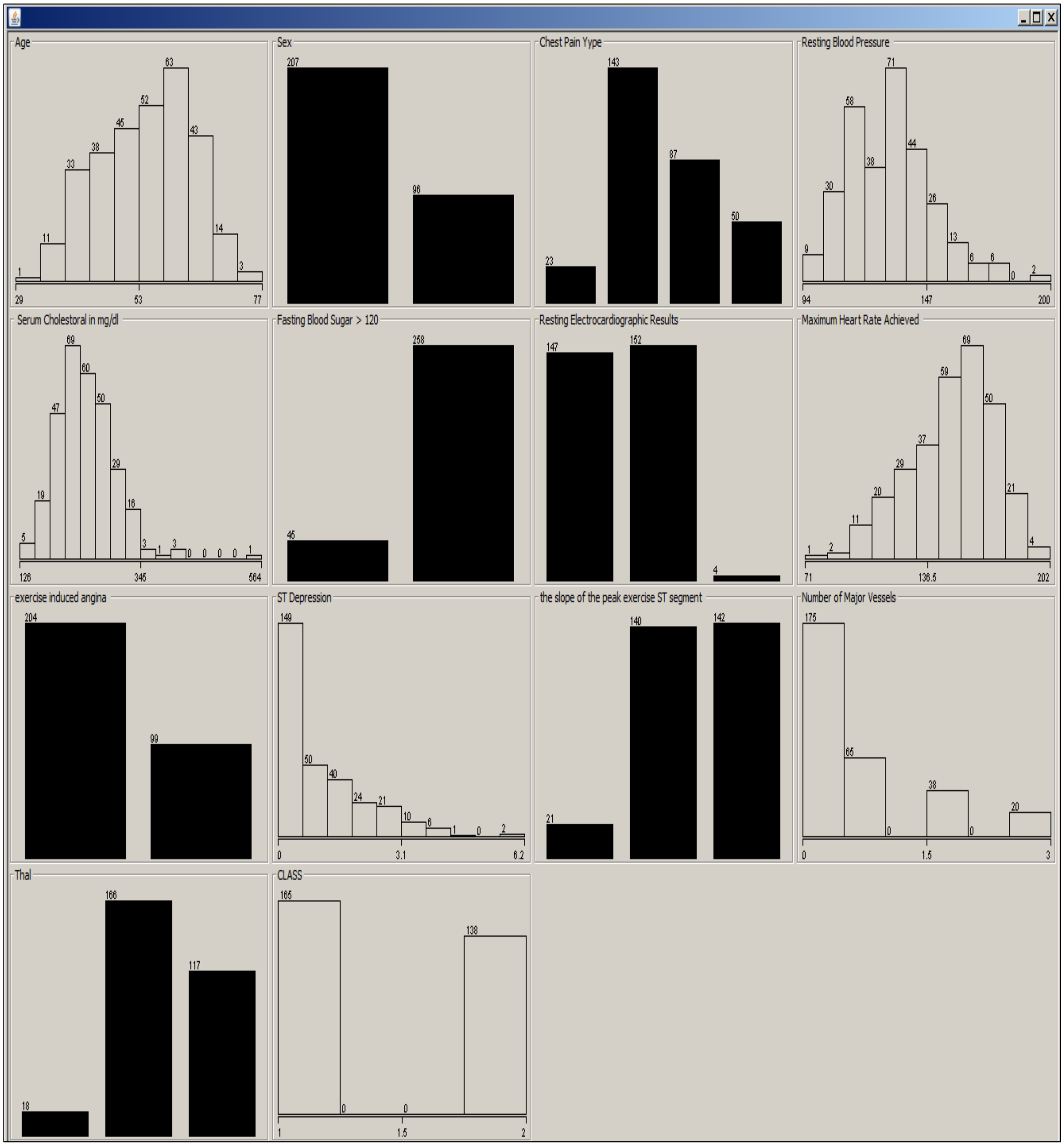

Age
Sex
Chest Pain Yype
Resting Blood Pressure
Serum Cholestoral in mg/dl
Fasting Blood Sugar > 120
Resting Electrocardiographic Results
Maximum Heart Rate Achieved
exercise induced angina
ST Depression
the slope of the peak exercise ST segment
Number of Major Vessels
Thal
CLASS

Figure 15: The data distribution of the Cleveland heart disease dataset

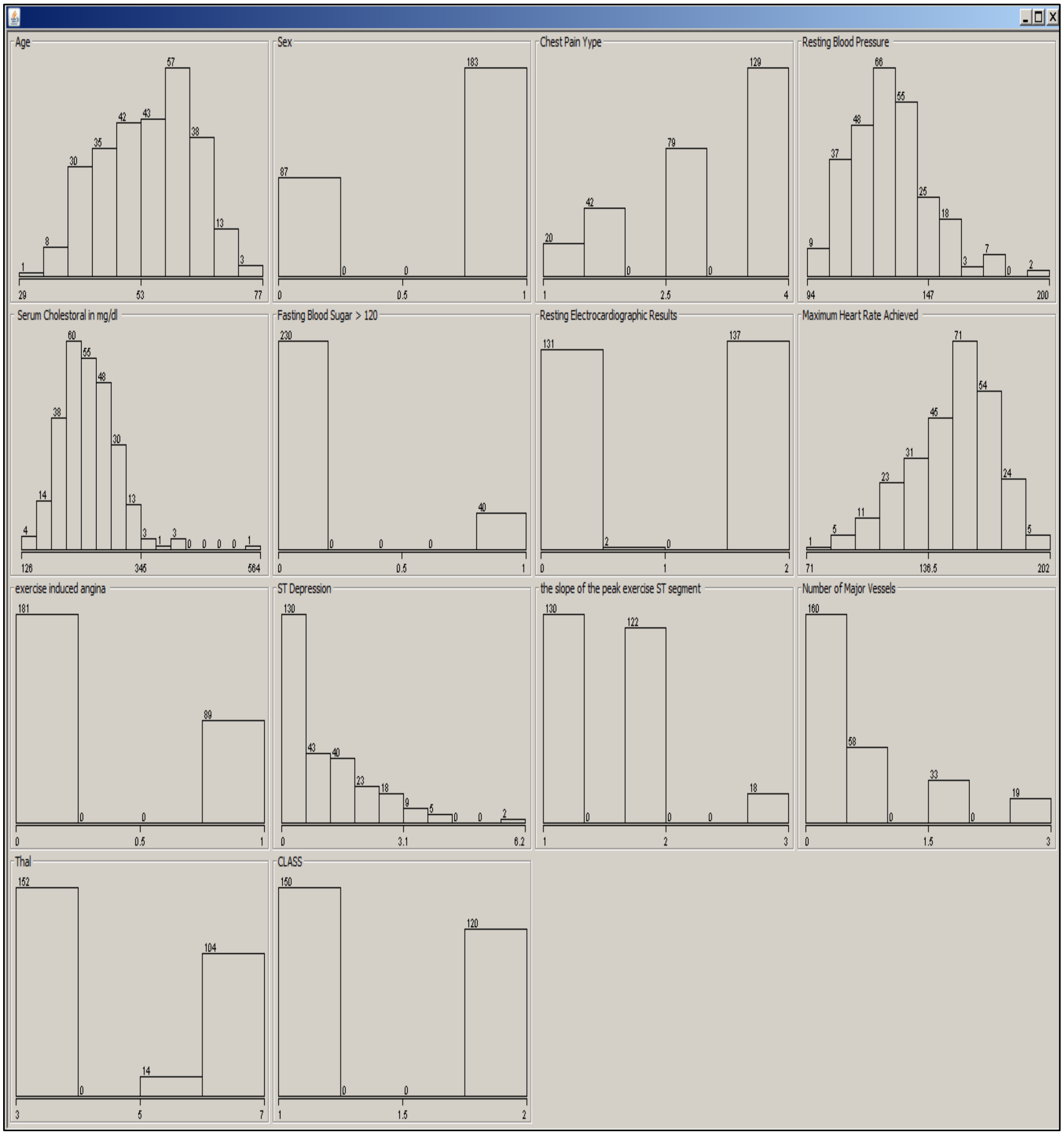


Figure 16: The data distribution of Statlog heart disease dataset

- **Credit Approval dataset**

The Credit Approval dataset contains 690 records, described by 6 numeric and 8 categorical attributes. The dataset is classified by human experts into two classes: the “positive” class with 307 objects, and the “rejected” class with 383 objects. Table 19 provides a discerption of this dataset. Note that the name of the attributes and the values are coded by the providers for security reasons.

Table 19 : Discerption of Credit Approval dataset

Attribute number	Attribute type	Distinct Values	Attribute Values / Range
1	nominal	2	a ,b
2	continuous	350	[13.75-80.25]
3	continuous	215	[0-28]
4	nominal	3	p ,g, gg
5	nominal	14	f, d ,I ,k ,j, aa ,m, c ,w, e, q, r, cc, x
6	nominal	8	ff ,dd ,j, bb ,v, n, o ,h, z
7	continuous	132	[0-28.5]
8	nominal	2	t ,f
9	nominal	2	t ,f
10	continues	23	[0-67]
11	nominal	2	t ,f
12	nominal	3	s, g, p
13	continues	171	[0-2000]
14	continues	240	[1-10001]

6.5. Evaluation of the Cluster Accuracy

6.5.1 Experiments and Results for Categorical Attribute Types

For categorical dataset, we compared our results with *k-mean for mixed data* and the *CLUC* algorithms.

- **Mushroom Dataset**

This dataset is classified into two clusters: poisonous cluster with 3916 objects and edible cluster with 4208 objects. With $\propto = 0.33$ *CLUC+* was able to assign correctly 3100 objects to the poison cluster and 4122 objects to the edible cluster. Table 20 shows the clustering results obtained from different clustering algorithms.

Table 20 : Relative performance of different clustering algorithms (Mushroom dataset)

Algorithm	Accuracy	Misclassification Error
CLUC+	92	0.08
CLUC	87	0.13
k- mean for mixed data	72	0.28

- **Congressional Votes Dataset**

This dataset was classified into two classes, Republicans class with 168 objects, and Democrats class with 267 objects. The *CLUC+* algorithm correctly assigned 160 objects to the Republicans cluster, and 221 objects to the Democrats cluster. Table 21 shows that the three algorithms generated closely similar results.

Table 21: Relative performance of different clustering algorithms (Vote dataset)

Algorithm	Accuracy	Misclassification error
CLUC+	88	0.12
CLUC	85	0.15
k- mean for mixed data	88	0.12

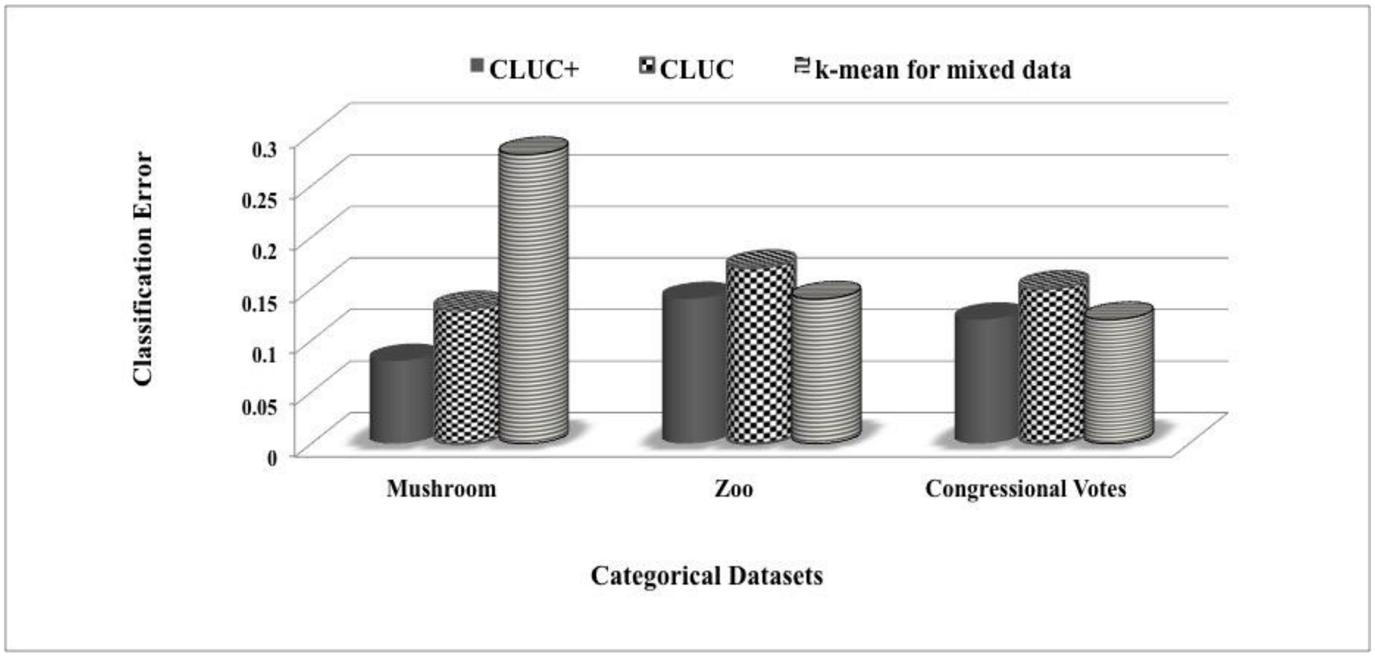


Figure 17: Clustering error for categorical datasets

6.5.2 Experimental Results on Numeric Attribute Types

- **Iris Dataset**

The dataset is classified into three classes each with 50 objects. We run *k-mixed* on Iris dataset with different *#B*, the best result was obtained when *#B* was set to 3 and $\propto$ was set to 0.30. Tables 22 illustrates the distributions of the objects over the three clusters. Table 23 shows the accuracy for different clustering algorithms.

Table 22 : Distribution of objects for Iris dataset with k-mixed

Cluster Number	Iris Setosa	Iris Versicolour	Iris Virginica
1	50	0	0
2	0	48	2
3	0	7	43

Table 23: Relative performance of different clustering algorithms (Iris dataset)

Algorithm	Accuracy	Misclassification error
k-mixed	94	0.06
k- mean for mixed data	93	0.07
Simple k-mean	88	0.12

- **Wine Dataset**

Human experts classified this dataset into 3 classes: class 1 with 59 objects, class 2 with 71 objects, and class 3 with 48 objects. With $\propto = 0.34$ and *#B* = 3, *k-mixed* assigned correctly 50 objects to the first clusters, 58 objects to the second cluster, and 48 objects to the third cluster. The results in Table 24 shows that our technique and simple k-mean show similar, low clustering accuracy.

Table 24: The Relative performance of different clustering algorithms (Wine dataset)

Algorithm	Accuracy	Misclassification Error
k-mixed	88	0.12
K-mean for mixed data	94	0.06
Simple k-mean	89	0.11

- **Breast Cancer Dataset**

The objects in this dataset are divided into two classes, benign class which consists of 458 objects, and malignant class which consists of 241 objects. The distinction of the values in this dataset is very small, compared to the other datasets we studied. Thus the best result was obtained when the number of bins *#B* was set to 10. That is, with $\propto = 0.30,$ *k-mixed* correctly assigned 457 objects to the binging class, and 220 objects to the malignant class. Table 25 shows the accuracy of different clustering algorithms.

Table 25 : Relative performance of different clustering algorithms (Breast Cancer dataset)

Algorithm	Accuracy	Misclassification Error
k-mixed	97	0.03
k- mean for mixed data	91	0.09
Simple k-mean	86	0.14

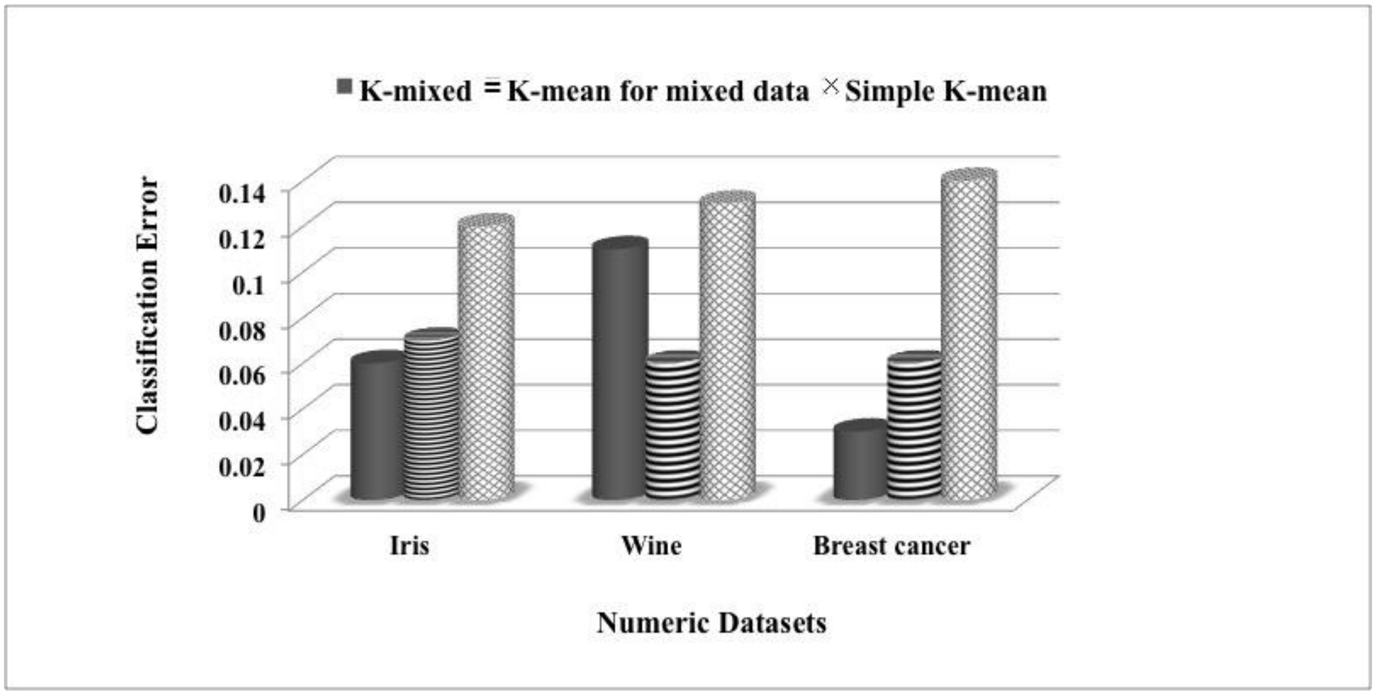


Figure 18: Clustering error for different numeric datasets

6.5.3 Experimental Results on Mixed Attribute Types

- **Credit Approval Dataset**

The objects in Credit Approval dataset are divided in two classes, class positive with 307 objects and class rejected with 383 objects. We applied *k-mixed* to credit approval dataset with $\propto = 0.43$ to produce two clusters. For this dataset, the best accuracy result was obtained when the number of bins *#B* is set to 3. That is, *k-mixed* correctly assigned 279 objects to the positive class, and 312 objects to the rejected class. For the *k-mean for mixed data* algorithm, we set the number of clusters to 2 and executed the algorithm 100 times. Table 26 shows the performance of *k-mixed* and *k-mean for mixed data*, and Figure 19 presents the performance of different algorithms. As can be seen from this table, our technique performed similarly to *k-mean for mixed data* and outperformed the two *Squeezer* and *k-prototype* algorithms.

Table 26 : Relative performance of different clustering algorithms (Credit Approval dataset)

Algorithm	Accuracy	Misclassification Error
k-mixed	86	0.14
k- mean for mixed data	86	0.14

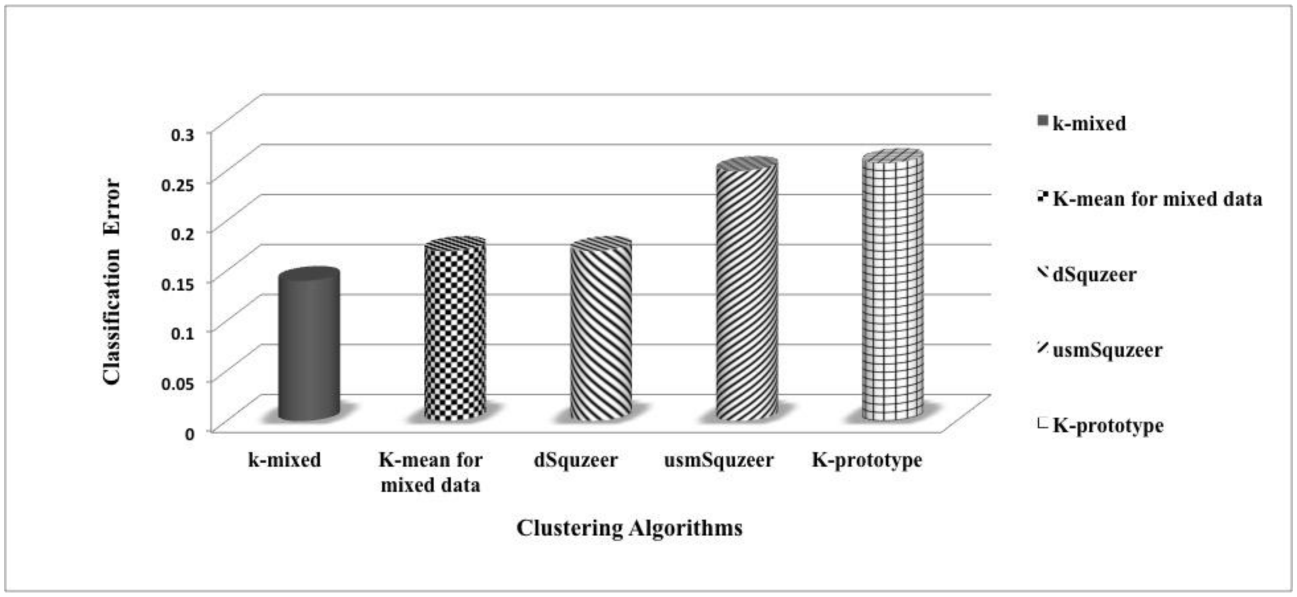


Figure 19: Clustering accuracy of different algorithms for credit approval datase

- **Herat Disease Datasets**

The Statlog dataset was classified into two class: presence class with 150 objects and absence class with 120 objects. *k-mixed* technique generates two clusters with $\propto = 0.34$. The best result was obtained when the number of bins *#B* is set to 5. For this, *k-mixed* correctly assigned 136 to the presence cluster, and 92 objects to the absence class. Table 27 represents the clustering accuracy obtained for *k-mixed* and *k-mean for mixed data*.

For the Cleveland dataset, we ran *k-mixed* with $\propto = 0.23$ and *#B* = 10. *K-mixed* succeeded to correctly assign 148 objects to the first class, and 110 to the second class. Table 28 shows the clustering accuracy for our technique and *k-mean for mixed data*, and Figure 20 shows the clustering accuracy for different algorithms for this dataset. The results indicate that *k-mixed* scored the highest accuracy for the two heart disease datasets.

Table 27 : Relative performance of different clustering algorithms (Statlog dataset)

Algorithm	Accuracy	Misclassification error
k-mixed	85	0.14
k- mean for mixed data	82	0.18

Table 28 : Relative performance of different clustering algorithms (Cleveland dataset)

Algorithm	Accuracy	Misclassification error
k-mixed	85	0.14
k- mean for mixed data	84	0.16

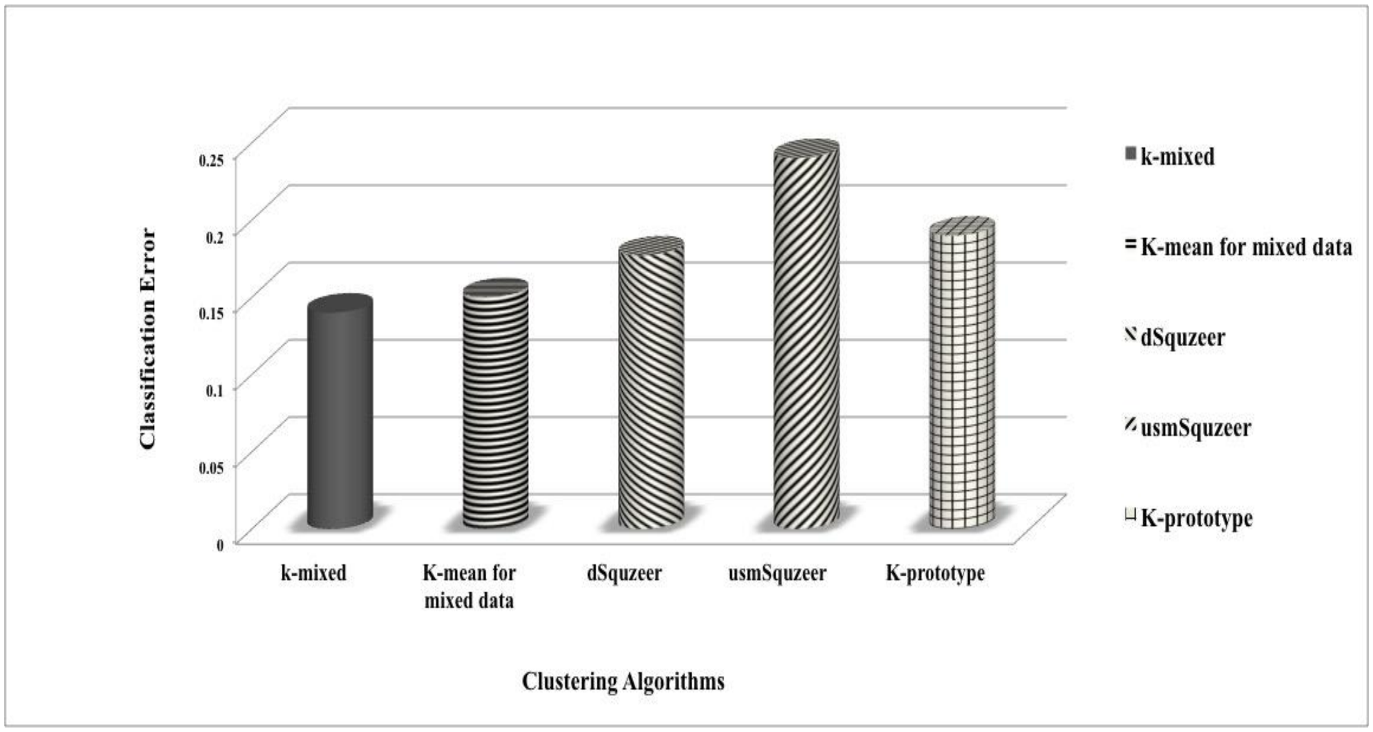


Figure 20: Clustering error of different algorithms for Cleveland heart disease dataset

6.6. Discussion

The results of our numerous experiments show that *k-mixed* provides better clustering results than *k-mean for mixed data* in two cases. The first case is when the variability (the number of distinct values) of the given categorical dataset is large (e.g., the Mushroom dataset). The second case in which this was observed is when the given datasets contains asymmetric binary attributes, e.g. the Cleveland and Statlog datasets.

For the first case, the similarity function of the proposed algorithm treat attributes values independent from each other. The clustering process is controlled by the values that considered as important, which get their weights based on their occurrences in the cluster. However, in the case of *k-mean for mixed dataset*, two values say *x* and *y* of an attribute *A* are considered "close" to each other if they have a strong connection. In other words, the clustering process is influenced by the values that co-occur with other attribute values. For the Mushroom dataset which is a "rich" dataset, i.e., contains many different values, compared to other datasets (see Table 11), such *x* and *y* values are less likely to be associated, which yields less accurate results.

For the second case in which the datasets has asymmetric attributes and the variety is small, our *k-mixed* algorithm providers better result because only important values were taken into account. For k-mean algorithm on mixed dataset, the two values x and y of the asymmetric attributes are more likely to co-occur and since this algorithm treats categorical values as one type, x and y get an equal, higher weight, which in turn leads to less accurate clustering result.

In the case of numeric datasets, the performance of k-mean algorithm on mixed dataset is affected by distribution of the data, whereas the performance of *k-mixed* depends on the number of bins considered in the discretization process. In our experiments, we observed that *k-mean for mixed data* produced more accurate clustering results than *k-mixed* when the given dataset has a normal distribution, like the Wine dataset (see Figure 21). For this dataset, our proposed technique generated less clustering result. If we look at the range of the attributes in Wine dataset (see Table 16), we can see that all but two of its attributes have the same scale (exceptions are magnesium and proline). We believe, the number of bins assigned to these attributes, which was the same as the other attributes, was not suitable, for causing a boundaries problem in which two close values were assigned to different intervals.

In the case of Breast Cancer dataset where the dataset has a positive/negative distribution (see Figure 22), we observed that k-mean for mixed data generated less accurate results. This can be attributed to the distance measure used by k-mean for mixed data (which was the Euclidian distance) in which high skewed data will throw off the mean and drastically alter covariance. Another factor which might influence the result is the strategy used in k-mean for mixed data algorithm to compute the weight for the numeric attributes, which depends on the discretization approach and the number #B is fixed to five regardless of the density degree of the attributes. For *k-mixed*, the best result was obtained when the number of bins was set to 10. Now if we consider Table 15, we can see that all its attributes are of the same scale and each has 10 numeric values. What should have been done in this case, we believe, is that the type of values in this dataset could be considered to be discrete in the sense that the values can be counted as distinct and separate. As a result, we obtained more accurate results when each value was assigned to a different interval.

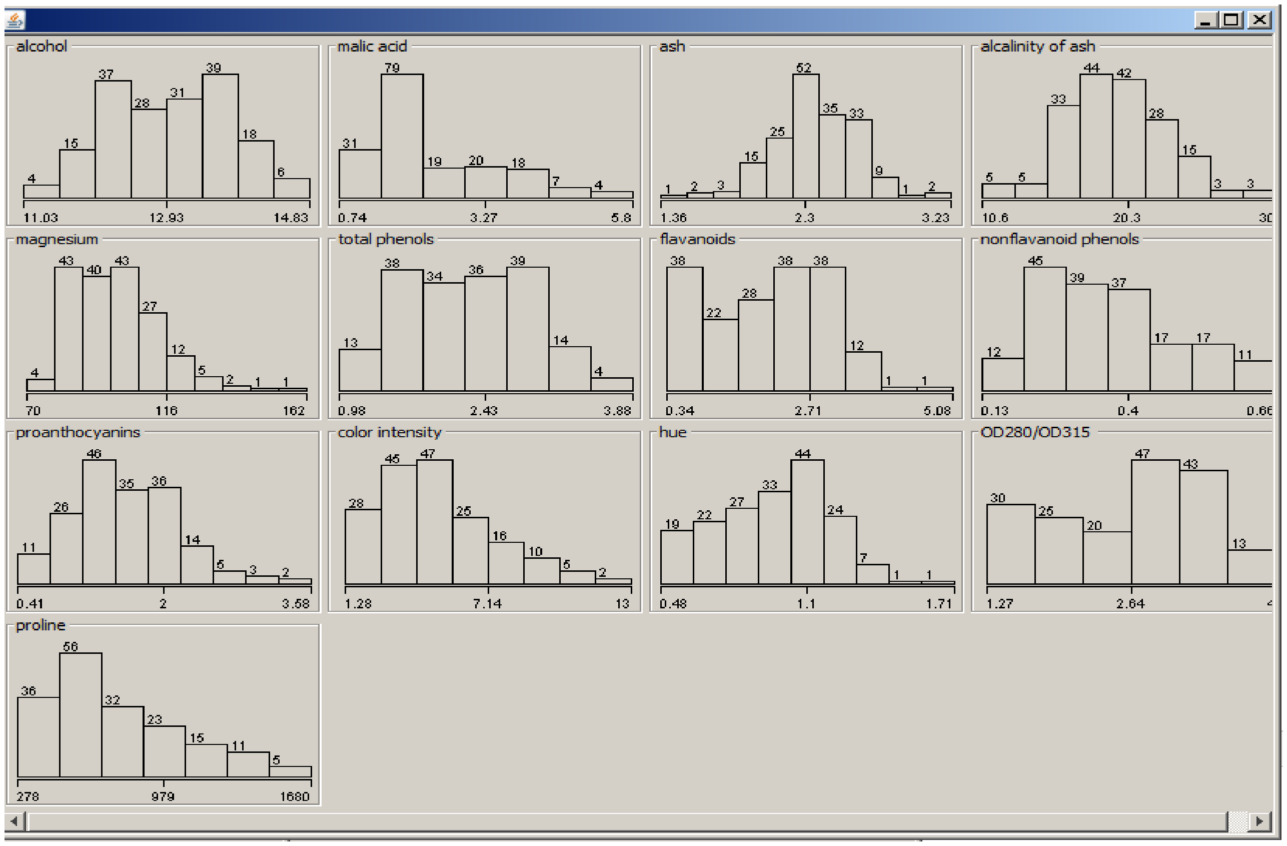


Figure 21 : Data distribution of Wine dataset

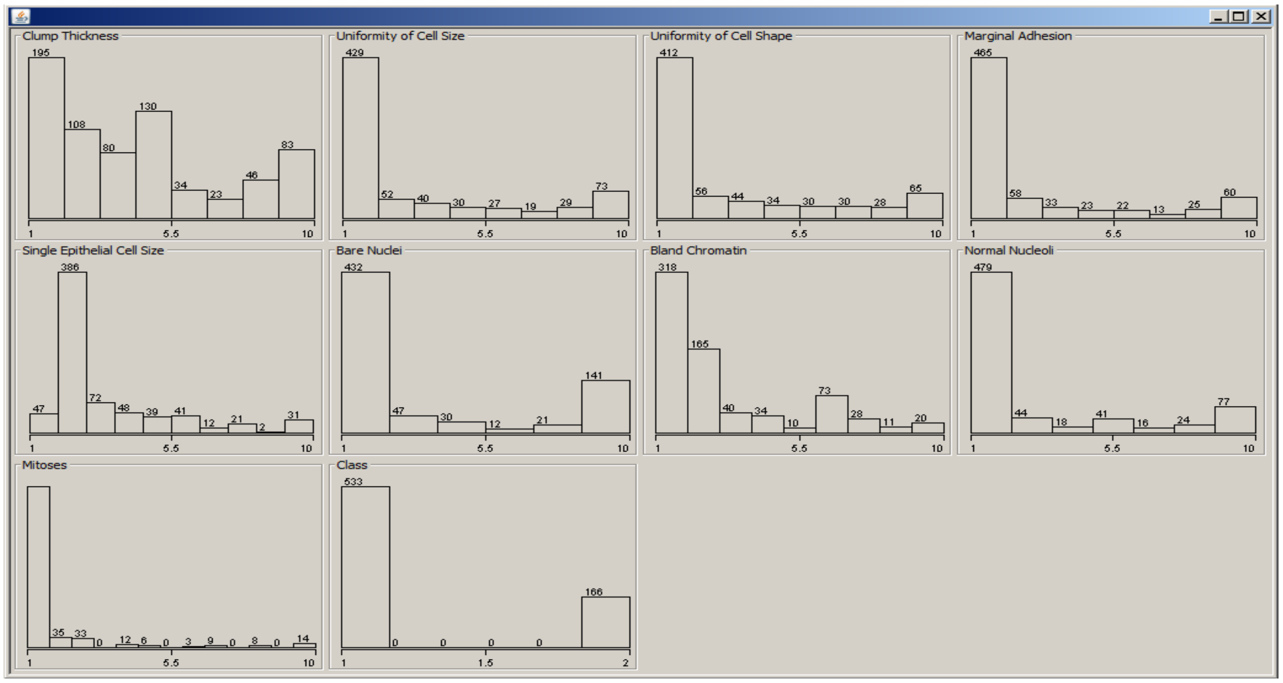


Figure 22 : Data distribution of Breast Cancer dataset

6.7. Robustness Evaluation

6.7.1 Insensitivity to input order

In this experiment we used two datasets, the Credit Approval dataset and the Cleveland heart disease datasets. For the Credit Approval dataset, we run *k-mixed* 20 times with different order randomly. In all the 20 times, *k-mixed* generated two clusters. In a few cases, we observed that a small number of tuples were assigned to different clusters, however these tuples did not much affect the final results. In most cases, the classification error obtained was the same as obtained with the original order; in one run the result was quite different with the classification error of 0.27.

For the Cleveland heart disease dataset, we run *k-mixed* 50 times. In all these 50 runs, two clusters were generated. In 47 times, we obtained the same classification error reported in Section 6.2.3, and in two runs our technique generated different results with the error 0.19 and 0.2, respectively.

For *k-mean for mixed data* algorithm, we did not test its sensitivity to the sequence order of the input objects, simply because this algorithm relies on random initialization values, that is, running the algorithm *n* times, gives *n* different solutions. Similarly, different input orders will generate different clustering results.

6.7.2 Detecting and Handling Outliers

We showed earlier in Chapter 4 that our technique can cope with outlier objects and placed them in separate clusters. In this section, we will examine our technique on a real-life dataset.

In our experiments, we used the Weka tool to identify the outlier objects in the dataset. Note that Weka uses the interquartile ranges method for filtering outliers. The dataset used for this experiment is leaf dataset, which consists of 340 objects each of which is described by 14 numeric attributes.

Weka identified 59 objects as outliers and 281 objects as normal. We run *k-mixed* on the dataset with different cohesion values in order to produce different clustering solutions. In each run, we analyzed the clustering results by examining the purity of clusters against outliers. Table 29 summarizes the results of our analysis and observations, which indicate that *k-mixed* is robust with respect to the order of the tuples in the input dataset and to outliers.

Table 29 : The clustering results generated by *k-mixed* algorithm

Cohesion	**Number of Clusters**	**No. Clusters without outliers**	**No.Clusters with outliers**	**No. Clusters (Outliers & normal)**	**Detected outliers**
0.77	61	32	15	14	30
0.80	67	34	13	11	35
0.87	149	111	34	4	51

6.8. Algorithm Efficiency

6.8.1 Handling Large Datasets

To investigate this, we run *k-mixed* and *k-mean for mixed dataset* on adult dataset (mixed attributes dataset), which consists of 48,842 objects, each of which describes by 14 attributes (6 numeric and 8 categorical). Adult dataset was classified by human experts based on the annual income into two clusters. Our algorithm succeeded to cluster the dataset in 30 seconds with $\propto = 0.29$. On the other hand, *k-mixed*, wasn't able to cluster the dataset and failed to construct even the initial clusters.

6.9. Summary

In this chapter, we studied the performance of our technique. We used different datasets and compared the performance of our proposed technique with some existing clustering algorithms. In terms of clustering quality, the experiments showed that our algorithm is effective for clustering not only on categorical or mixed datasets, but also promising to generate good clustering results on pure numeric datasets. The results obtained with *k-mixed* in most cases were quite similar to *k-mean for mixed data* algorithm and outperformed others algorithms. In a few cases, the performance of our algorithm was slightly below that of *k-mean for mixed data* algorithm.

In terms of robustness and handling large datasets, our technique demonstrated to be effective in coping with outliers and the order of tuples in the input data. Moreover, and unlike *k-mean for mixed data*, *k-mixed* has the ability to cluster dataset with large number of object with a small memory. This indicates more promise for better scalability of our proposed algorithm to handle larger datasets.

.

Chapter 7. Conclusion and Future Work

7.1. Summary and Conclusion

Most of the existing clustering algorithms for mixed data look at the data generally as numeric and categorical, ignoring the fact that categorical attributes could be either nominal or binary. We proposed a clustering technique called *k-mixed* that makes this distinction and also considers that binary attributes are of two different types and their contributions should also be different during the clustering process.

In order to evaluate our technique, we conducted numerous experiments using different real-life, benchmark datasets. The results have shown that our solution resulted in improved quality of the generated clusters and efficiency for reduced number of iterations required for the clustering process to converge. The experiments also showed that while discretization is an affordable pre-processing step, it generally leads to generation of better results than the combined clustering method, as we observed for the two Squeezer algorithms. Our proposed clustering *k-mixed* enjoys the following properties:

- It inherits the advantage of *CLUC* algorithm, in which the number of clusters is not a predefined parameter, rather it is generated automatically from the data itself. Also, *k-mixed* has the ability to cluster high dimensional datasets. *k-mixed* does not require maintaining the entire dataset in the main memory but rather requires keeping only some information about the clusters.
- *k-mixed* is a robust technique. The clustering result is insensitive to the input order of the objects, and can cope with outliers by placing them in separate clusters.

- *k-mixed* can be applied to different types of datasets effectively: numeric, categorical, and mixed dataset.

Despite the above strengths of *k-mixed*, it has the following two limitations and drawbacks:

1. Its output quality depends on the number of intervals during the discretization phase. Improper number of bins could negatively affect the quality of the generated clusters.
2. It was observed that when the number of clusters is large, *k-mixed* generates poor quality clusters.

7.2. Future work

Although the proposed technique has shown good results, its performance could be improved by ideas and techniques to overcome the aforementioned two short comings. In one direction, we could investigate more sophisticated discretization techniques in the pre-processing step, as opposed to using the simple one we used in this work. Another future work is investigating ways to improve the efficiency of the algorithm when the number of “expected” clusters are large.

REFRENCES

[1] J. Hsn, M. Kamber, and J. Pei, Data Mining concepts and techniques, Morgan kaufmann Publishers, 2006.

[2] G. Gan, C. Ma, and J. Wu, Data Clustering: theory,algorithms, and applications, vol. 20, Siam, 2007.

[3] J. MacQuuen, "Some methods for classification and analysis of multivariate observation," *in: Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability*, p. 281–297, 1967.

[4] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *KDD,* vol. 69, pp. 226-231, 1996.

[5] G. Karypis, E.H. Han, V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer,* vol. 32(8), pp. 68-75, 1999.

[6] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," *In ACM SIGMOD,* vol. 27, pp. 73-84, 1998.

[7] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Proceedings of 15th International Conference on Data Engineering*, pp. 512-521, 1999.

[8] A. Nemalhabib, and N. Shiri, "CLUC: a natural clustering algorithm for categorical datasets based on cohesion," *In Proceedings of the 2006 ACM symposium on Applied computing,* pp. 637-638, 2006.

[9] The university of Walikato, "Weka," [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[10] "UCI machine learning repository," [Online]. Available: https://archive.ics.uci.edu.

[11] J.Wang, in *Data warehousing and mining : concepts, methodologies, tools, and applications,* USA, Information Science Reference, 2009, pp. 240-248.

[12] Q. Zhao, "Cluster validity in clustering methods.," Publications of the University of Eastern Finland., 2012.

[13] S. Pandit, and G. Suchita, "A comparative study on distance measuring approaches for clustering," *nternational Journal of Research in Computer Science,* vol. 2(1), pp. 29-31, 2011.

[14] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *ACM SIGMOD,* vol. 25, pp. 103-114, 1996.

[15] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery,* vol. 2.3, pp. 283-304, 1998.

[16] H. Liu, F. Hussain, C.L. Tan, and M. Dash, "Discretization: An enabling technique," *Data mining and knowledge discovery,* vol. 6(4), pp. 393-423, 2002.

[17] C. Li, and G. Biswas, "conceptual clustering with numeric and nominal mixed data - a new similarity based system," *IEEE Transactions on Knowledge and Data Engineering,* 1998.

[18] R. Ghaemi, MN. Sulaiman, H. Ibrahim, and N. Mustapha, "A survey: clustering ensembles techniques," *World Academy of Science, Engineering and Technology,* vol. 50, pp. 636-645, 2009.

[19] D Kumar , "Divide and ConquerMethod for Clustering Mixed Numerical and Categorical Data.," *International Journal of Computer Science and Information Technologies,* vol. 4(1), pp. 103-106, 2013.

[20] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," *In Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD),* pp. 21-34, 1997.

[21] A. Ahmad, and L. Dey, "A k-mean clustering algorithm for mixed numeric and categorical data," *Data and Knowledge Engineering,* vol. 63(2), pp. 503-527, 2007.

[22] Y. M. Cheung, and H. Jia, "A Unified Metric for Categorical and Numerical Attributes in Data Clustering," *Advances in Knowledge Discovery and Data Mining,* pp. 135-146, 2013.

[23] F. Cao, J. Liang, D Li, L. Bai, and C. Dang, " A dissimilarity measure for the k-Modes clustering algorithm," *Knowledge-Based Systems,* vol. 26, pp. 120-127, 2012.

[24] C. Li ,and G. Biswas, "conceptual clustering with numeric and nominal mixed data - a new similarity based system," *IEEE Transactions on Knowledge and Data Engineering,* 1998.

[25] D.W. Goodall, "A New Similarity Index Based On Probability," *Biometrics,* pp. 882-907, 1966.

[26] R. A.Fisher, Statistical methods for research workers, Edinburgh and London : Oliver and Boyd, 1963.

[27] H. Lancaster, "The combining of Probabilities arising from data in discrete distributions.," *Biometrika,* vol. 36, pp. 370-383, 1949.

[28] Z. He, X. Xu, and S. Deng, "Scalable algorithms for clustering large datasets with mixed type attributes.," *International Journal of Intelligent Systems,* vol. 20(10), pp. 1077-1089, 2005.

[29] Z. He, X. Xu, and S. Deng, "Squeezer: an efficient algorithm for clustering categorical data," *Journal of Computer Science and Technology,* vol. 17(5), pp. 611-624, 2002.

[30] Y. Cheung, and H. Jia, "Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number," *Pattern Recognition,* vol. 46(8), pp. 2228-2238, 2013.

[31] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *In: Proc Fourth Int Conf on Knowledge Discovery and Data Mining,* New York, 1998.

[32] M. Y. Shih, J. W. Jheng, and L. F. Lai, "A two-step method for clustering mixed categorical and numerical data," *Tamkang Journal of Science and Engineering,* vol. 13(1), pp. 11-19, 2010.

[33] S. Sayad, "An introduction to Data Mining," [Online]. Available: http://www.saedsayad.com/unsupervised_binning.htm.

[34] "The R Project for Statistical Computing," [Online]. Available: http://www.r-project.org/.

[35] MVJ. Reddy, and B. Kavitha, "Efficient ensemble algorithm for mixed numeric and categorical data," *IEEE International Conference on Computational,* 2010.

[36] He Z, Xu X, and Deng. S, "Clustering mixed numeric and categorical data: A cluster ensemble approach," *arXiv preprint cs/0509011,* 2005.

[37] J. Suguna, and A. M. Selvi, "Ensemble fuzzy clustering for mixed numeric and categorical data," *Int. J. Comput. Appl,* vol. 42, pp. 19-23, 2012.

[38] D. Kumar, "Divide and ConquerMethod for Clustering Mixed Numerical and Categorical Data.," *International Journal of Computer Science and Information Technologies,* vol. 4(1), pp. 103-106, 2013.

[39] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," *In Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD),* pp. 21-34, 1997.

[40] D. W. Goodall, "A New Similarity Index Based On Probability," *Biometrics,* pp. 882-907, 1966.

[41] R. A. Fisher, Statistical methods for research workers, Edinburgh and London: Oliver and Boyd, 1963.

[42] H. Lancaster, "The combining of Probabilities arising from data in discrete distributions.," *Biometrika*, vol. 36, pp. 370-383, 1949.

[43] Z. He, X. Xu, and S. Deng, "Scalable algorithms for clustering large datasets with mixed type attributes.," *International Journal of Intelligent Systems*, vol. 20(10), pp. 1077-1089, 2005.

[44] M. Y. Shih, J. W. Jheng, and L. F. Lai, "A two-step method for clustering mixed categorical and numerical data," *Tamkang Journal of Science and Engineering*, vol. 13(1), pp. 11-19, 2010.

[45] J.MacQuuen, "Some methods for classification and analysis of multivariate observation," *in: Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability*, p. 281–297, 1967.

[46] Z.Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2.3, pp. 283-304, 1998.

[47] R. Xu, and D. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions*, vol. 16(3), pp. 645-678, 2005.