DOMINANT ATTRIBUTE AND MULTIPLE SCANNING APPROACHES FOR DISCRETIZATION OF NUMERICAL ATTRIBUTES

BY

H. SHANKER RAO

Submitted to the graduate degree program in Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

_____

Chairperson: Dr. Jerzy W. Grzymala-Busse

_____

Dr. Perry Alexander

_____

Dr. Doina Caragea

Date Defended: September 8, 2014

The Thesis Committee for H. SHANKER RAO

certifies that this is the approved version of the following thesis:

DOMINANT ATTRIBUTE AND MULTIPLE SCANNING APPROACHES FOR DISCRETIZATION OF NUMERICAL ATTRIBUTES

_____

Chairperson: Dr. Jerzy W. Grzymala-Busse

Date approved: September 8, 2014

ABSTRACT

Rapid development of high throughput technologies and database management systems has made it possible to produce and store large amount of data. However, making sense of big data and discovering knowledge from it is a compounding challenge. Generally, data mining techniques search for information in datasets and express gained knowledge in the form of trends, regularities, patterns or rules. Rules are frequently identified automatically by a technique called *rule induction*, which is the most important technique in data mining and machine learning and it was developed primarily to handle symbolic data. However, real life data often contain numerical attributes and therefore, in order to fully utilize the power of rule induction techniques, an essential preprocessing step of converting numeric data into symbolic data called *discretization* is employed in data mining.

Here we present two entropy based discretization techniques known as *dominant attribute approach* and *multiple scanning approach*, respectively. These approaches were implemented as two explicit algorithms in a JAVA programming language and experiments were conducted by applying each algorithm separately on seventeen well known numerical data sets. The resulting discretized data sets were used for rule induction by LEM2 or *Learning from Examples Module 2* algorithm. For each dataset in *multiple scanning approach*, experiments were repeated with incremental scans until interval counts were stabilized. Preliminary results from this study indicated that *multiple scanning approach* performed better than *dominant attribute approach* in terms of producing comparatively smaller and simpler rule sets.

ACKNOWLEDGEMENTS

It has been an honor for me to have a very special thesis committee for my Master's degree. Their support at various time points through the course of my study is invaluable and I am quite sure that without their care, affection, help and guidance I wouldn't have completed my degree.

First, I would like to express my gratitude and extend my sincere thanks to Dr. Jerzy W. Grzymala-Busse for agreeing to be my thesis advisor. He has been a source of inspiration to me and I really enjoyed learning data mining skills from him. I had taken all the courses he teaches at The University of Kansas and I was convinced that I had met a gold-mine of knowledge. I was initially amazed with his knowledge, simplicity and ability to present complex concepts in a simplified manner, however, that also made me apprehensive for quite some time in approaching him to request to be my advisor. He has been always extremely focused in his approach, very much accessible for a friendly discussion and spent ample amount to time in understanding my thought process. He taught me how to systematically approach and tackle a research problem. Besides his guidance through research, I cannot forget the little big things he does to students in general and in particular to me to ensure that I succeed and complete thesis in a timely manner.

I would like to extend thanks to Dr. Doina Caragea for extending her mentorship to me even after I transfer from Kansas State University to The University of Kansas and agreeing to be my thesis committee member. I had taken a Bioinformatics course she teaches in Kansas State University and at that time it was unbelievable for me to find someone as elegant, knowledgeable, skillful and honest as herself. Right after the completion of the course, I requested her to become my thesis committee member and thereafter we consolidated upon that

relationship till date. I am very grateful for her invaluable guidance, friendship and for becoming one of the very special persons in my life.

I would also like to thank Dr. Perry Alexander for agreeing to be my thesis committee member. I had taken a Programming languages course taught by Dr. Alexander in the initial phase of my Master's degree. I was making a difficult transition from Bioinformatics / Molecular genetics to Computer Science and I wasn't sure of myself succeeding in the endeavor. Even though my research focus was not directly related to Dr. Alexander's research, his nice demeanor and friendly attitude allowed me to discuss with him frankly my difficulties in adjusting in the new environment. His help, support and encouragement to carry on helped me surpass initial phase. I wish I was more involved in research activities with him.

Also, I am extremely thankful to Dr. Dongkyu Choi for giving me financial support in the form of graduate assistantship and letting me gain valuable experience in his Cognitive Control System Laboratory. I am also thankful for the financial assistance and guidance I received at various time points from Dr. Gerald H. Lushington, Dr. Russ Waitman and Dr. Peter G. Smith that helped me explore and pursue research career in the field of Computational biology and Data mining.

Finally, I would like to thank my family, friends and academic & non-academic staff at The University of Kansas for making my studies a success. And my sincere apologies for missing out many many names here that have, overtime, contributed in me.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Machine learning, data mining and expert systems are interrelated subfields of artificial intelligence. One of the primary objectives in artificial intelligence is to make the intelligent agent learn rules from data automatically [1]. Whereas *machine learning* equips machine the ability to learn by recognizing patterns present in training data and superimpose inferences later on unseen data [2], *data mining* is defined as extraction of hidden, previously unknown, and potentially useful high-level information from low-level data [3]. *Expert systems* are used to implement specific domains of expertise where knowledge is represented in the form of rules and reasoned in a given scenario by testing their applicability by induction or deduction [4]. These special kind of computer programs have a wide scope in commercial, industrial and scientific applications.

Real life data exhibit varied structure and there exist numerous data mining techniques, however, no single technique can be considered the best that would be applicable on all scenarios. Often raw data needs to be cleansed and transformed to make it suitable for data mining and knowledge discovery.

Many real life applications involve data that are in *numeric* format, however, most of the inductive learning algorithms, including the one used in this thesis, require data to be in *symbolic* format. In order to use such rule induction algorithms, numeric data must be converted into a symbolic format and the process of this conversion is known as discretization.

Since entropy based methods are regarded as superior among several existing discretization methods, we present here two improved entropy based discretization methods viz. *dominant attribute approach* and *multiple scanning approach* [5, 6]. Dominant attribute approach is a

purely recursive algorithm, where after each cycle, data set is split into subsets based on the dominant attribute only and recursion continues until a stopping criterion is satisfied. On the other hand, in multiple scanning approach, all attributes are simultaneously scanned for a fixed number of times and if the stopping criterion is not yet satisfied, dominant attribute algorithm is invoked to complete discretization. In both approaches, continuous attributes are initially converted into discrete intervals and later some of the neighboring intervals are merged together. The merging algorithm preserves consistency by implementing merge process in two steps: (a) *Safe merging* – neighboring intervals are merged if all instances of them are labeled by the same decision value; and (b) *Proper merging* – neighboring intervals are merged only if the result of merging do not reduce level of consistency. Seventeen well known data sets, frequently used in data mining experiments were chosen to test our discretization algorithms.

CHAPTER 2. BACKGROUND

Discretization of numerical attributes is one of the basic preprocessing techniques used in data mining. Many discretization algorithms have been proposed, however, discretization based on entropy is regarded as best. Before embarking upon entropy based discretization, we introduce here the basic concepts of data mining, rough set theory, probability theory and information theory.

2.1. Knowledge discovery & data mining

In statistics, the study of dependence is called regression. The goal is to summarize the observed data as simply, usefully and elegantly as possible [7]. Regression analysis aims to construct a suitable model by employing mathematical rigor on a small sample. The process is usually slow and conclusions, expressed only in terms of statistical errors, lack explanation. On the other hand, modern *data mining* (DM) process is fast, adventurous and explores entire population by using powerful algorithms. It provides better explanation of results in terms of rule sets, decision trees, graphs, support vectors, etc., while the predictive power of various algorithms is tested in terms of confusion matrix on unseen data. *Knowledge Discovery in Databases* (KDD) is an automatic, exploratory analysis and modeling of large data repositories. KDD is the organized process of identifying valid, novel, useful, and understandable patterns from large and complex data sets. DM is the core of KDD process, providing algorithmic infrastructure of rule induction and inference engine to the overall knowledge acquisition framework. KDD is an iterative and interactive process summarized in following steps [8]:

1. ***Understanding of the application domain***: In this preparatory phase, the investigator gathers information, understands the problem and defines goals. In the process, data miner makes up

understanding of consequences of various choices to be made during data cleansing, preprocessing, data mining and post-processing phases.

2. ***Selecting and creating a data set***: Having understood the problem and set goals for problem solving, next step is to collect and organize data for knowledge discovery. The data from varied sources is obtained and integrated into a common knowledgebase.
3. ***Preprocessing and cleansing***: Integration of raw data collected from one or more sources may not be straight-forward. Real data is often marred with errors, missing values and technician bias. Preprocessing and cleansing phase is the opportunity to normalize, remove or mitigate inconsistencies and enhance reliability of data significantly.
4. ***Data transformation***: After the initial cleansing phase, data may be free from intrinsic flaws but it may not be suitable for intake into the favorite data mining algorithm. Transformation is the process of converting raw data into a form that is better suited for rule induction in the targeted algorithm. Some of the frequently used methods include discretization, dimension reduction, transforming dependent variable only, independent variables only or both kind of variables simultaneously, etc.
5. ***Choosing the appropriate Data Mining task***: Data mining may mean a different thing to different people. Sometimes simple statistical analysis is sufficient whereas in other occasions even a very sophisticated algorithm is not sufficient. Data mining may be broadly subdivided into a problem of regression analysis, cluster analysis or classification. Depending on project needs, investigator may choose a suitable data mining strategy.
6. ***Choosing the Data Mining algorithm***: Having the broad strategy, next step is to decide on the finer tactics. Many algorithms have been developed to solve the same problem and in data mining too, different algorithms can achieve the same goal with different trade-offs. For

example, classification problem can be addressed by rule induction, generation of decision trees, construction of neural networks, support vector machines, etc. whereas clustering problem can be addressed by techniques of nearest-neighbor, K-means, hierarchical clustering, etc. Each of the methods have some advantages and disadvantages and depending on the availability of resources in terms of time, money and effort, investigator makes a conscious choice of a particular method to be pursued.

7. ***Employing the Data Mining algorithm***: The selected data mining algorithm is implemented and various parameters are tuned to suit the datasets under investigation.
8. ***Evaluation***: Performance of selected algorithms is evaluated on the experimental data sets. This is usually done by a process called *n-fold cross validation* and summarizing outcome in the form of *confusion matrix*. Confusion matrix comprise of 2×2 matrix where each slot is occupied by the computed value variously known as *true positive*, *false positive*, *false negative* and *true negative* respectively. Greater proportion of true positives and true negatives imply worthiness of the algorithm.
9. ***Using and maintaining the discovered knowledge***: The knowledge becomes active when the implemented system is brought outside of the experimental environment and tested on practical situations. Sustaining effectiveness in varied conditions determines robustness of the implemented methodology. Providing periodic updates and implementing patches are important components of any maintenance program.

2.2. Decision table

Data from which rules are induced are presented in the form of a table, in which *cases* and *attributes* are represented by rows and columns respectively. An example of such table is

presented in Table 1. The last column usually represents a dependent variable called *decision* that contains expert assigned values whereas all other columns are independent variables called *attributes*. The set of all cases is denoted by U and the set of all attributes is denoted by A. *Decision* is denoted by d, and comprised of *concepts*. All cases in a particular concept are labelled by the same decision value.

Table 1. Decision table

		ATTRIBUTES				**DECISION**
		A_1	A_2	…	A_n	d
CASES	1	v_{11}	v_{21}	…	v_{n1}	d_1
	2	v_{12}	v_{22}	…	v_{n2}	d_2
	3	v_{13}	v_{23}	…	v_{n3}	d_3
	…	…	…	…	…	…
	m	v_{1m}	v_{2m}	…	v_{nm}	d_m

2.3. Rough set theory

In the seminal work on rough set theory, Z. Pawlak [9] made a clear distinction between rough sets and classic sets. In classic set theory, sets were described as precise entities that are bound by crisp boundaries and uniquely determined by its elements. However, many concepts in nature are vague and since they are often associated with entities in the boundary region, understanding of vagueness is critical in decision making. In order to overcome this limitation of classic sets, Z. Pawlak introduced the concept of rough sets where imprecision is expressed by a boundary region between sets. Crux of the theory prescribes to split universe into lower and upper approximations. The *lower approximation* represents a subset of elements that certainly belong to the concept whereas the *upper approximation* represents a subset, in which some of the elements certainly

belong to the concept and others possibly belong to the concept. All other elements certainly do not belong to the concept (Figure 1).

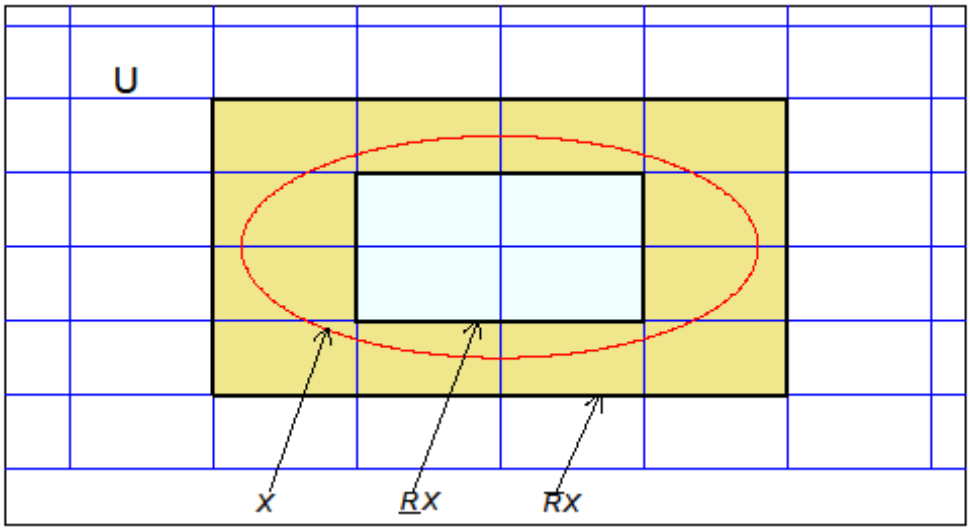

Figure 1. Rough sets

Let U be a nonempty set of elements called the *universe*. For any set A of attributes, an indiscernibility relation $R(A)$ is defined for any two cases $x, y \in U$ by

$$(x, y) \in R(A) \text{ if and only if } a(x) = a(y) \text{ for any } a \in A,$$

where $a(x)$ is the value of the attribute a for the case x. *Indiscernibility relation* represents uncertainty associated with elements in U. The *indiscernibility relation* R is an equivalence relation. An equivalence class, called an elementary set, and determined by any $x \in U$, is denoted by $[x]_R$. Let $X \subseteq U$ and in order to characterize X with respect to R, rough set theory introduced the following concepts:

- ***Lower approximation*** of a set X with respect to $R(A)$ is the set of all elements which can be for certain classified as X with respect to R (or certainly in X)

$$\underline{R}X = \bigcup_{x \in U} \{[x]_R | [x]_R \subseteq X\}.$$

- ***Upper approximation*** of a set X with respect to $R(A)$ is the set of all elements which can be possibly classified as in X (or possibly in X in view of $R(A)$)

$$\overline{R}X = \bigcup_{x \in U} \{[x]_R | [x]_R \cap X \neq \emptyset\}.$$

- ***Boundary region*** of a set X with respect to R is the set of all elements, which can be classified neither as X nor as *not-X* with respect to R

$$RN_R X = \overline{R}X - \underline{R}X.$$

Set X is considered *rough* if the *boundary region* is nonempty, otherwise *crisp*.

2.4. Rule induction

Regularities hidden in the data are usually expressed in the form of rules and rule induction is one of the most important techniques of machine learning and data mining [10]. For the decision table shown in Table 1, let $A = \{a_1, a_2, .., a_n\}$ be a set of attributes, and let $\{v_1, v_2, .., v_n\}$ be a set of corresponding values, and $d = \{c_1, c_2, .., c_k\}$ a set of *decision values*. A *block of attribute-value pair*, $[(a, v)]$ is a set of all cases with identical v in a:

$$[(a, v)] = \{x | a(x) = v\}$$

Similarly, a *block of decision values*, $[c]$ is a set of all cases with identical c in d:

$$[c] = \{x | d(x) = c\}$$

Patterns in the data are expressed in the form of a rule set. A single rule is a combination of one or more (a_i, v_j) pairs and (d_{c_x}) such as:

$$(a_1, v_1)\ and\ (a_2, v_2)\ and\ \dots and\ (a_n, v_n)\ then\ (d_{c_x})$$

or

$$(a_1, v_1)\ \&\ (a_2, v_2)\ \&\ \dots \&\ (a_n, v_n)\ \rightarrow\ (d_{c_x})$$

Any *attribute-value pair* in the left hand side of a rule is called *condition* part and the right hand side is called a *decision-value* for the rule. If a rule induction algorithm explores set of all attribute values, it is considered as *global* whereas if exploration is confined only to a set of certain attribute-value pairs, it is called *local*.

1. ***Global covering***: Let $A = \{a_1, a_2, .., a_n\}$ and $d = \{c_1, c_2, .., c_n\}$ be sets of *attributes* and *decision values*, respectively. The equivalence classes of indiscernibility relation $R(A)$ are called A-elementary sets and denoted by $[x]_A$. A partition on U constructed from all $[x]_A$ will be denoted by A^*. For decision variable, $\{d\}$-elementary sets are called *concepts*, and the corresponding partition is denoted as $\{d\}^*$.

   The simplest approach to rule induction is based on finding the smallest subset B of the set A that is sufficient to be used in a rule set. A partition B^* is smaller than or equal to partition $\{d\}^*$ if and only if for each block P of B^* there exists a block P' of $\{d\}^*$ such that $P \subseteq P'$. The relation is expressed as $B^* \leq \{d\}^*$, and called *attribute dependency inequality*. For a decision d we say that $\{d\}$ depends on B if and only if $B^* \leq \{d\}^*$, i.e., for any B-elementary set $[x]_B$, there exists a *concept* C from $\{d\}^*$ such that $X \subseteq C$. A global covering of $\{d\}$ is a subset B of A such that $\{d\}$ depends on B and B is minimal in A.

   The algorithm to compute a single global covering is implemented as LEM1 (*Learning from Examples Module, version 1*) algorithm and described in [10-13]. The LEM1 algorithm is based on calculus on partitions on the entire universe U.

2. ***Local covering***: LEM2 algorithm (*Learning from Examples Module, version 2*) [10-13] presents another approach to rule induction where search space is limited to attribute-value pairs only. Let T be a set of attribute-value pairs. The *block* of T, denoted by $[T]$, is the following set

$$\bigcap_{t \in T} [t]$$

Let B be a subset of d. Set B depends on a set T of attribute-value pairs $t = (a, v)$ if and only if $[T]$ is nonempty and $[T] \subseteq B$. Set T is a minimal complex of B if and only if B depends on T and no proper subset T' of T exists such that B depends on T'. Let $\mathcal{T}$ be a nonempty collection of sets of attribute-value pairs. Then $\mathcal{T}$ is a local covering of B if the following conditions are satisfied:

(1) Each member T of $\mathcal{T}$ is a minimal complex of B,

(2) $U_{t \in \mathcal{T}} [T] = B$, and

(3) $\mathcal{T}$ is minimal, i.e., $\mathcal{T}$ has the smallest possible number of members.

2.5. Probability theory

Practical data mining often deals with data sets that are noisy, inconsistent or incomplete and therefore rules induced from such data sets are associated with certain amount of uncertainty. Probability theory is the calculus of uncertainty and it is a key concept in the field of data mining and knowledge discovery. Some of the basic terms used in probability theory are briefly described below [14]:

1. ***Random variable***: A random variable is a variable selected at random from a statistical population. If a random variable has a finite number of possible values, it is called a *discrete random variable*, for example, number of students in a class, number of eggs in a basket, etc. If possible values of a random variable are continuous, it is called a *continuous random variable*, for example, height of students, temperature in℃, etc.

2. ***Probability***: The probability of an event E is defined as the ratio of number of favorable outcomes, N_e to the total number of possible outcomes N.

$$P(E) = \frac{N_e}{N}$$

3. ***Conditional probability***: For the two chance events E_1 and E_2, not necessarily independent, conditional probability of E_1 given E_2 is defined as the ratio of occurrence of both events, E_1 and E_2 together to the occurrence of E_2 irrespective of E_1.

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}$$

4. ***Probability distribution***: Probability distribution of a discrete random variable is a set of probabilities associated with each of its possible values. For instance, consider a random variable *Color* with a domain {*green*, *yellow*, *yellow*, *red*, *blue*, *red*, *yellow*}. Probability associated with each value in *Color* is computed in Table 2 and the distribution is displayed in Figure 2.

Table 2. Probability distribution

Random variable	**Probability**
green	1/7 = 0.14
yellow	3/7 = 0.43
red	2/7 = 0.29
blue	1/7 = 0.14

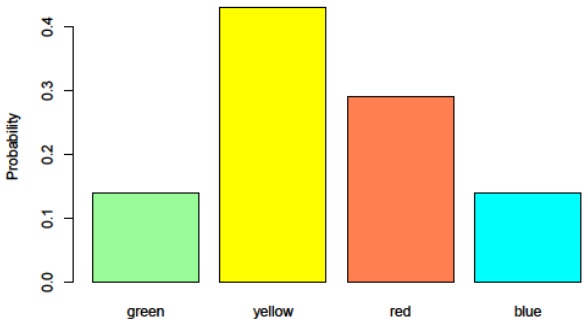

Figure 2. Probability distribution

Similar treatment with continuous random variable is problematic because it is impossible to assign small amount of probabilities to all possible values in a continuous random variable. To overcome this problem, its range is divided into a number of intervals and respective probabilities are computed as the number of cases falling into those defined intervals. If we increase the number and decrease the width of intervals, resulting probability distribution becomes almost a smooth curve.

5. ***Cumulative probability distribution***: Cumulative probability of a value is the sum of probabilities of all values up to itself in the ordered list and cumulative probability distribution is the set of all cumulative probabilities for possible values in the random variable. Table 3 shows respective cumulative probabilities for the values in Table 2.

Table 3. Cumulative probability distribution

Random variable	Probability	Cumulative probability
green	0.14	0.14
yellow	0.43	0.57
red	0.29	0.86
blue	0.14	1.00

Probabilities for continuous random variables are computed as the area under a curve and the total area under the curve is equal to 1.

2.6. Information theory and entropy

Information theory started as a subfield to communication theory and primarily addressed issues with data compression and data communication. However, its domain has grown and made significant contributions to other fields of study such as statistical physics, computer science, statistical inference, probability, etc. [15]. *Entropy*, *relative entropy* and *mutual information* are the fundamental quantities of information theory and are defined in terms of probability

distributions. These concepts were first formulated and introduced in relation to communication theory by C. E. Shannon [16]. They characterize behavior of random variables by quantifying amount and rate of information produced by the random processes.

Let a random process generates n possible events with probabilities of $p_1, p_2, \dots, p_n$ respectively. The *entropy* of such a variable is defined by:

$$H(X) = -\sum_{i=1}^{n} p_i \,.\log p_i$$

This quantity measures randomness or uncertainty associated with the variable. For example, the quantity vanishes for a completely certain event and measures high for highly uncertain event i.e., there are more choices with equally likely events. The quantity plays a central role in information theory as it provides measures of information, choice and uncertainty. Entropy of X, denoted by $H(X)$, has following properties:

1. $H(X) = 0$ if and only if all but one p_i are zero and the sole non-zero probability is equal to unity. Thus entropy vanishes only when the outcome of a particular event is certain. Otherwise it has a positive value.
2. For a given n, $H(X)$ is maximum and equal to $log\ n$ when all the p_i are equal. This is the most uncertain situation.
3. Let x and y are two random variables with m and n possible outcomes respectively. Let p_{ij} be the probability of the joint occurrence of i^{th} and j^{th} instance of x and y respectively. *Marginal entropies* of two variables are defined by:

$$H(x) = -\sum_{i=1,j=1}^{m,n} p_{i,j} \,.\log \sum_{j=1}^{n} p_{i,j}$$

$$H(y) = -\sum_{i=1,j=1}^{m,n} p_{i,j} \,.\log \sum_{i=1}^{m} p_{i,j}$$

It can be easily observed that $H(x,y) \leq H(x) + H(y)$. This imply that the uncertainty of a joint event is always less than or equal to the sum of the individual uncertainties.

4. Any change toward equalization of the probabilities $p_1, p_2, \ldots, p_n$ increases $H(X)$.
5. For the random variables x and y, *conditional entropy* of y given x is defined as the average of the entropy of y for each value of x, weighted according to the probability of getting that particular x:

$$H(y|x) = -\sum_{i=1,j=1}^{m,n} p_{i,j}.\log p_{j|i}$$

Where $p_{j|i}$ is the conditional probability of p_j given p_i. Conditional entropy measures average uncertainty of y when x is known.

CHAPTER 3. DISCRETIZATION

Discretization is a family of data transformation techniques in which continuous numerical values are transformed into a finite set of discrete intervals. For a numerical attribute A with an interval $[a, b]$ as range, discretization of A is defined as a partition of the range into n intervals:

$$\{[a_0, a_1), [a_1, a_2), \dots, [a_{n-2}, a_{n-1}), [a_{n-1}, a_n]\}$$

where $a_0 = a$, $a_n = b$, and $a_i < a_{i+1}$ for $i = 0, 1, \dots, n-1$. The numbers $a_1, a_2, \dots, a_{n-1}$ are called *cut-points*. Discretization methods are called *local* if attributes are processed one at a time and *global* if all attributes are simultaneously considered towards selection of a best *cut-point*. A comprehensive review of discretization methods can be found in [6, 17-19].

3.1. Equal width intervals

This is the simplest kind of discretization technique where entire range is partitioned into a number of equal width intervals. According to H. A. Sturges [20], for an attribute A with N cases and range $a_N - a_1$, optimal class intervals CL, can be estimated from the formula:

$$CL = \frac{a_N - a_1}{1 + 3.332 \log N}$$

This method can be used for computation of basic summary statistics of frequency distributions, however, it does not take into account the class information and it generally fares poor during rule induction processes. With equal width interval methods, it is difficult to determine the optimal number of intervals and often the optimal count is settled by running the learning algorithm iteratively on same data set but with incremental interval count on each iteration. The process is cumbersome and the determined number may not be optimal.

3.2. Equal frequency intervals

Another simple approach where interval widths may vary but sample frequency is same in every interval and therefore all discretized intervals have equal information content. Again, the desired number of intervals must be determined stochastically or supplied by the user.

3.3. Minimal class entropy method

The method computes class entropy associated with subsets of values partitioned by the selected cut-point. Let class C has k concepts associated with a set S, then *class entropy* of S, $E(S)$ is defined as:

$$E(S) = -\sum_{i=1}^{k} \frac{|c_i|}{|S|} \log_2 \frac{|c_i|}{|S|}$$

where $|S|$ and $|C_i|$ are the cardinalities of S and i^{th}concept respectively. Negative sign in the expression assures that the quantity is always positive, whose lower value implies closer association (or better fit) between set and class. To evaluate a cut-point q for an attribute A, weighted average of class entropies $E(A, q; S)$ of the partitioned subsets S_1 and S_2 are determined as:

$$E(A, q; S) = \frac{|S_1|}{|S|} E(S_1) + \frac{|S_2|}{|S|} E(S_2)$$

This quantity is called the *class information entropy* [21]. Binary discretization for an attribute is determined by computing $E(A, q_i; S)$ for all possible cut-points and selecting the one for which the quantity is minimum. The process is recursively applied to the subsets until a stopping criteria is satisfied. *Minimum description length principle* (MDLP) criteria is one of such

approaches that accepts cut-point if the result of partition leads to a positive information gain, otherwise recursion in the discretization process stops without further partitioning [22].

3.4. Cluster analysis method

Cluster analysis is frequently used for unsupervised machine learning where class information is not taken into consideration. The main idea is to compute one-to-one distances among all samples and partition them accordingly into a number of clusters. Again, deciding on the optimal number of clusters is an iterative process that can be determined from a number of different approaches. Often the process is more of an art than science and it is often swayed by an expert's predispositions. Cluster analysis based discretization [18] described here uses *level of consistency* as the stopping criterion during cluster formation stage. A^* and $\{d\}^*$ represent partitions on U constructed from A and d, respectively. The *level of consistency*, $L(A)$ is defined as:

$$L(A) = \frac{\sum_{X \in \{d\}^*} |\underline{A}X|}{|U|}$$

A desired value for *level of consistency* is unity after discretization. Therefore, stopping condition of recursion in the binary discretization algorithm is $L(A) = 1$. Recursion prevails as long as $L(A) < 1$. The discretization process consists of two distinct steps, (a) *cluster formation* and (b) *post-processing*. Each of the steps are briefly described below:

1. ***Cluster formation***: If there are m samples and n numeric attributes, all attributes are normalized and $m \times m$ distance matrix is constructed. The choice of distance measure affects clustering and therefore it should be chosen carefully. In the agglomerative technique of clusters analysis, initially every sample is treated as a single cluster and the two closest

clusters are fused together. Fused cluster is treated as a single entity and its centroid of is used to re-compute distances from remaining clusters. Consistency of clusters is computed by a rough set approach and fusion process is repeated until the level of consistency denoted by L_c is preserved to the original state. In rough set theory, data with a set of samples U and a set of attributes A is consistent with respect to the decision d, if and only if $A^* \leq \{d\}^*$, where A^* and $\{d\}^*$ are partitions on U constructed from A and d respectively.

2. ***Post processing***: Cluster formation often induce excessive intervals, some of which are fused together during the post-processing step. Some of the neighboring intervals are merged together in such a way that the consistency of resulting clusters is preserved to the original state. Let the neighboring intervals are denoted by $i..j$ and $j..k$, then merging them together results in a new interval $i..k$. The merging algorithm consists of two steps, safe merging and proper merging. (a) *Safe merging*: Neighboring intervals are merged if all instances of them are labeled by the same decision value. (b) *Proper merging*: Neighboring intervals are merged if the result of merging do not reduce level of consistency.

3.5. Entropy based discretization

Entropy based discretization takes into consideration the information content of both attribute and decision variables and therefore it is considered as one of the most successful approach. We present here two improved entropy based discretization strategies viz. *dominant attribute approach* and *multiple scanning approach* [5, 6].

3.5.1. Dominant attribute approach

1. ***Identify best attribute***: Best attribute is the one which has highest *information gain*. Given decision d, information gain $I(a)$ associated with an attribute a is defined as:

$$I(a) = H_d(U) - H(d|a)$$

where $H_d(U)$ is the entropy of d and $H(d|a)$ is the conditional entropy of d given a.

2. ***Identify best cut-point***: For the best attribute, sort the values and enumerate all possible cut-points. Find out the best cut-point which has lowest *class information entropy*.
3. ***Split dataset***: The best cut-point splits dataset S (initially S is equal to U) into two smaller datasets, S_1 and S_2.
4. ***Stopping criteria***: Compute level of consistency $L(A)$ of the best cut-point and if $L(A) < 1$, apply steps 1 through 3 recursively to subsets S_1 and S_2 separately.
   If $L(A) = 1$, recursion stops and binary discretization for a particular subset is complete.

3.5.2. Multiple scanning approach

1. ***Total number of scans***: The parameter denoted by t, must be provided by the user.
2. ***Identify best cut-points***: Let a set of numerical attributes is denoted by A. Scan the entire dataset and find out best cut-point for every attribute in A. For each attribute separately, sort the values and enumerate all possible cut-points. Find out the best cut-point which has lowest *class information entropy*.
3. ***Level of consistency***: Discretize all attributes in A with best cut-points and denote a new set of discretized attributes by A^D. Compute level of consistency $L(A^D)$ and if $L(A^D) < 1$, compute partition $(A^D)^*$ on U.
4. ***Split dataset***: For $\forall_{x \in (A^D)^*}$, if $x \not\preceq \{d\}^*$ extract subset S with all elements of x. If number of scans is less than t, apply recursively steps 1 through 4 for subset S.
5. ***Stopping criteria***: The algorithm stops when the number of predefined scans are

exhausted and level of consistency is preserved to 100%.

$$Number\ of\ scans = t; and$$

$$L(A^D) = 1$$

If t has exhausted but $L(A^D) < 1$, apply dominant attribute algorithm to the remaining subsets.

3.6. Post processing

Excess intervals produced by dominant attribute approach and multiple scanning approach are handled by post processing procedure described earlier for cluster analysis based discretization. Briefly again, some of the neighboring intervals are merged together in such way that the number of intervals are reduced and at the same time level of consistency is preserved. Let the neighboring intervals are denoted by $i..j$ and $j..k$, then merging them together results in a new interval $i..k$. The merging algorithm consists of two steps:

a) ***Safe merging***: Neighboring intervals are merged if all instances of them are labeled by a same decision value.

b) ***Proper merging***: Neighboring intervals are merged if the result of merging do not reduce level of consistency.

CHAPTER 4. IMPLEMENTATION

4.1. Computer platform

All experiments were run on a machine located in the Eaton Hall laboratory, EECS department, University of Kansas. Machine configuration included 8 GB of RAM with 64 bit processor (Intel(R) Xeon(R) CPU E3-1270 V2 @ 3.50GHz) under Fedora (Linux) operating system.

4.2. Programming language

The algorithms were implemented in Java programming language using Eclipse integrated development environment (IDE) Kepler service release 1. Java was originally developed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems, Inc. in 1991 [23]. The main impetus for the development of Java was to liberate the language from platform dependence and although heavily inspired from C++, Java was never meant to replace C++ (Figure 3). Java is a platform independent language and therefore once written, it can be run anywhere.

Some of the salient features of Java include:

1. ***Simple***: Java was designed to be easy to learn and use effectively. Complex operations such as handling memory leaks and garbage collection are taken care by automatic memory management and thus all the complexities are hidden from the programmer. Since Java inherits syntax and object-oriented features of C++, many C++ programmers find it rather simple to learn Java.
2. ***Platform independent***: Both system software and machine architecture have been evolving continuously and therefore, one of the challenges for programmers is to maintain their own

code for execution on different platforms and at different times. Java allows program to be written once and run anywhere/anytime.

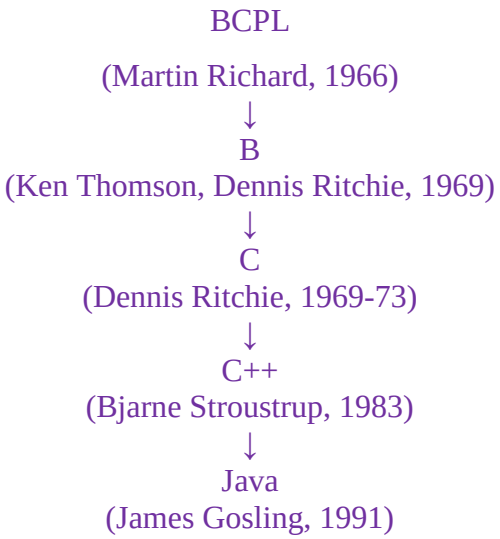

Figure 3. Evolution of Java

3. ***Bytecode***: Java achieved platform independence by implementing *bytecode* and *Java Virtual machine* (JVM). The Java compiler processes source code and generate bytecode. Bytecode is different from usual executable code and it is highly optimized for JVM. JVM creates a layer between native platform and the bytecode. Since the upper layer of JVM is always same, a bytecode can be run on a wide variety of platforms.
4. ***Secure***: Java restricts internet based applets to its own execution environment and therefore other system resources are protected from unauthorized access.
5. ***Easy to distribute via internet***: Java handles TCP/IP protocols and therefore its applications can be easily transmitted via internet.
6. ***Industry standard***: Platform independence give a big advantage to any industry and the automatic memory management hides unnecessary technical jargon.

4.3. Graphics

All graphics were generated by using suit of plotting packages implemented in R programming languages (*R version 3.1.0 (2014-04-10) -- "Spring Dance"*) [24] with RStudio version 0.98.953 integrated development environment (IDE).

4.4. Data structures

A complex problem can be divided into a number of sub-problems and the solution can be reached in a different ways. Algorithms must be written to maximize the chances of achieving goal and minimize the amount of time and effort involved. The efficiency issue becomes most obvious when the size of input data is large. As an example, a poorly written algorithm for maximum subsequence sum takes 2.28 seconds for input size of 1000 but it fails to come up with solution for larger dataset of size 10,000. On the other hand, same problem can be solved in 0.0003 seconds with efficient algorithm [25]. Keeping-up with the earlier discussion, data mining algorithms are highly complex, exploration intensive and goal oriented. Specific choices made in the course of action has profound impact on the quality of results.

Java provides a convenient facility for using desired data structures [26]. The *java.util* package contains a powerful subsystems called collections which is Java's standard framework of handling group of objects. The framework has highly efficient implementations of various fundamental data structures such as arrays, linked lists, trees, hash tables, etc. Some of the data structures used in this thesis work and respective running times in big O notation [27] is briefly summarized below.

1. ***List interface***: List is a sequence of elements where duplicates are allowed and ordering is not important. Elements in the list are accessed by their position and elements at specific

position can be inserted or removed.

- *LinkedList*: It provides a bidirectional linked-list data structure. It has two constructors, the first builds a head (empty linked list) and the second builds a linked list on it. Because every node has to maintain two links, Java's LinkedList is a very inefficient implementation. Average running times for insert, delete and search operations are $O(1)$, $O(1)$ and $O(n)$ respectively.
- *ArrayList*: In Java, ArrayList supports dynamic arrays that is created with an initial size and it can automatically grow or shrink during run time. Average running times for insert, delete and search operations are $O(n)$, $O(n)$ and $O(1)$ respectively.

2. ***Set interface***: A set does not allow duplicate elements.
   - *TreeSet*: It uses tree data structure where objects are stored in sorted, ascending order. Average running times for insert, delete and search operations are $O(log\ n)$, $O(log\ n)$ and $O(log\ n)$ respectively.
   - *HashSet*: It uses hash table for storage. Hash table stores information by using a mechanism called hashing. In hashing, the informational content of a key is used to determine a unique value, called its hash code. The hash code is then used as the index at which the data associated with the key is stored. The advantage of hashing is that it allows the execution time of basic operations to remain constant. Average running times for insert, delete and search operations are $O(1)$, $O(1)$ and $O(1)$ respectively.
3. ***Map interface***: A map is an object that stores associations between key/value pairs. The key must be unique, but the values may be duplicated.
   - *TreeMap*: TreeMap implements the map interface by using a tree. A TreeMap provides an efficient means of storing key/value pairs in sorted order and allows rapid retrieval. A

TreeMap guarantees that its elements will be sorted in ascending key order. Average running times for insert, delete and search operations are $O(log\ n)$, $O(log\ n)$ and $O(log\ n)$ respectively.

- *HashMap*: The HashMap class uses a hash table to implement the Map interface. This allows the execution time of basic operations to remain constant. The order in which elements are added to a hash map is not necessarily the order in which they are read by an iterator. Average running times for insert, delete and search operations are $O(1)$, $O(1)$ and $O(1)$ respectively.

4. ***Iterator***: Often it is necessary to cycle through the elements in a collection. Every collection class implements an iterator with similar interface and therefore, elements of any collection class can be accessed through the methods defined in the iterator. In other words, iterator interface gives a general-purpose, standardized way of accessing the elements within a collection.
5. ***Loops***: In addition to well established data structures, due care was taken while looping through procedures. For example, if the requirement was just to iteration through list, *while* loop was preferred over *for* loop. FOR loop has an overhead of computing list size and increment operator. Because nesting has exponential cost on the algorithms, nested loops were avoided whenever possible.

4.5. Data sets

Data sets used to conduct experiments are summarized in Table 4 and most of them are available in the *Machine Learning Repository*, University of California Irvine. Number of cases, attributes and classes for each data set along with pointers to source information is included in the table.

Table 4. Data sets

Data set	Number of		
	Cases	**Attributes**	**Concepts**
Australian Credit Approval [28] (AUSTR)	690	14	2
NCBI GEO number: GSE2564 [29] (COMMON-COMBINED-LERS)	68	16280	11
M-BANK[30]	66	5	2
Echocardiogram [28] (M-ECHO)	74	7	2
Glass Identification [28] (M-GLASS)	214	9	6
M-GLOBE[28]	33	5	4
Image Segmentation [28] (M-IMAGE)	210	19	7
Iris [28] (M-IRIS)	150	4	3
Wine [28] (M-WINE)	178	13	3
Abalone [28] (N-ABALONE)	4177	8	28
Liver Disorders [28] (N-BUPA)	345	6	2
Ecoli [28] (N-ECOLI)	336	7	8
Pima Indians Diabetes [28] (N-PIMA)	768	8	2
Waveform Database Generator [28] (N-WAVE-512)	512	21	3
PRICE[6]	7	3	5
TABLE[5]	7	3	5
TRIP[12]	8	3	2

4.6. Instructions for running software

The software can be run from the directory containing java source code. It is convenient to create a project directory and execute program from there. From the project directory, typing ***make*** will compile, link and execute program and on-screen instructions will guide user through rest of the

program.

1. CD to project directory
2. Type '*make*' to execute program (program will then invoke user to enter other particulars)
    a. input file name
    b. number of scans
    c. whether to save list of cutpoints
    d. output file name – discretization
    e. if response of (c) is 'y', provide name for cutpoints file
    f. output file name - safe merging
    g. output file name - proper merging

CHAPTER 5. EXPERIMENTS

Algorithms implemented for dominant attribute approach and multiple scanning approach were applied separately on all 17 data sets. Stopping criteria for dominant attribute approach was to preserve level of consistency equal to 100%. For multiple scanning approach, experiments were repeated with incremental scan counts. As the number of scans increases, fluctuations with discretized interval counts gradually decreases and when the fluctuation was no longer significant, the number of scans was considered optimal. The resulting discretized data sets were then used to induce rules using LEM2 algorithms implemented in the LERS data mining system. To clarify things, we describe here worked out example of each algorithm by using a data set shown in Table 5:

Table 5

CASE	ATTRIBUTE			DECISION
	weight	length	height	Price
1	0.8	0.3	7.2	very small
2	0.8	1.1	7.2	Small
3	0.8	1.1	10.2	Medium
4	1.2	0.3	10.2	Medium
5	1.2	2.3	10.2	Medium
6	2.0	2.3	10.2	High
7	2.0	2.3	15.2	very high

5.1. Dominant attribute algorithm

We illustrate this method by using data set shown in Table 5.

A. ***Find best attribute***. Dominant attribute is the one which results in maximum information gain, where *information gain* is given as: $I(a) = H_d(U) - H(d|a)$. In this expression, since

entropy of decision do not change, we infer that information gain is maximum for the attribute which has minimum *conditional entropy*. Conditional entropy of the decision *d* is defined as:

$$H(d|a) = -\sum_{j=1}^{m} p(a_j).\sum_{i=1}^{n} p(d_i|a_j).\log p(d_i|a_j)$$

where $a_1, a_2, .., a_m$ are all values of a and $d_1, d_2, .., d_n$ are all *concepts* in d. The computed values of conditional entropies of respective attributes in Table 5 is:

H(price|weight) = 0.965

H(price|length) = 1.25

H(price|height) = 0.745*

Minimal conditional entropy is associated with *height*.

B. ***Find best cut-point***. Next step is to find the best cut-point for *height*. To enumerate all potential cut-points, we first sort unique values in the attribute and find mid-point between the adjacent values. For *height*, the potential cut-points are 8.7 and 12.7:

Potential cut-points = 7.2→**8.7**←10.2→**12.7**←15.2

To evaluate a cut-point q in a variable V, weighted average of class entropies $E(V, q;\ S)$ of the partitioned subsets S_1 and S_2 are determined as:

$$E(V, q; S) = \frac{|S_1|}{|S|} E(S_1) + \frac{|S_2|}{|S|} E(S_2)$$

The computed values of $E(A, q;\ S)$ for each cut-point is:

E(*height*, 8.7, *U*) = 1.265*

E(*height*, 12.7, *U*) = 1.536

Since minimal value is associated with the 8.7, we consider it as the best *cut-point*.

C. ***Level of consistency***. The level of consistency, $L(B)$ is defined as:

$$L(B) = \frac{\sum_{X \in \{d\}^*} |\underline{B}X|}{|U|}$$

Level of consistency for the partitioned data set across cut-point, $height_{8.7}$, $L(B) = 0$. Since $L(B) < 1$, partition Table 5 at cut-point $height_{8.7}$, and repeat steps A, B and C recursively with both subsets, Table 6 and Table 7.

Table 6

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8	0.3	7.2	very small
2	0.8	1.1	7.2	small

Table 7

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
3	0.8	1.1	10.2	medium
4	1.2	0.3	10.2	medium
5	1.2	2.3	10.2	medium
6	2.0	2.3	10.2	high
7	2.0	2.3	15.2	very high

Consider Table 6.

A. ***Find best attribute***. Computed values of conditional entropies for three attributes:

$$H(price|weight) = 1$$

$$H(price|length) = 0^*$$

$$H(price|height) = 1$$

Minimal conditional entropy is associated with *lenght*.

B. ***Find best cut-point***. Only potential cut-point for length is 0.7:

Potential cut-points = 0.3→**0.7**←1.1

C. ***Level of consistency***. *Level of consistency* for the partitioned data set across cut-point, $length_{0.7}$, $L(B) = 1$. Since level of consistency is 100%, stopping criterion has been satisfied.

Consider Table 7

A. ***Find best attribute***. Again, we compute conditional entropy for each attribute:

$$H(price|weight) = 0.4^*$$

$$H(price|length) = 0.95$$

$$H(price|height) = 0.65$$

Minimal conditional entropy is associated with *weight*.

B. ***Find best cut-point***. For *weight*, potential cut-points and conditional entropy associated with each cut-point is computed as:

Potential cut-points = 0.8→**1.0**←1.2→**1.6**←2.0

Conditional entropy,

$$E(weight, 1.0, U) = 1.2$$

$$E(weight, 1.6, U) = 0.4^*$$

Since minimal value is associated with the 1.6, we consider it as the best *cut-point*.

C. ***Level of consistency*** at cut-point, $weight_{1.6}$, $L(B) = 0.6$. Since $L(B) < 1$, split Table 7 at cut-point $weight_{1.6}$, and repeat steps A, B and C recursively with the resulting subsets, Table 8 and Table 9.

Table 8

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
3	0.8	1.1	10.2	medium
4	1.2	0.3	10.2	medium

5	1.2	2.3	10.2	medium

Table 9

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
6	2.0	2.3	10.2	high
7	2.0	2.3	15.2	very high

Consider Table 8. Decision value of all cases in Table 8 are identical, which means the data set is consistent and therefore we do not need to discretize it any further.

Consider Table 9

A. ***Find best attribute***. Computed values of conditional entropy for three attributes are:

$$H(price|weight) = 1$$

$$H(price|length) = 1$$

$$H(price|height) = 0^*$$

Minimal conditional entropy is associated with height.

B. ***Find best cut-point*** for height

Potential cut-points = 10.2⟶**12.7**⟵15.2

C. ***Level of consistency*** at cut-point, $height_{12.7}$, $L(B) = 1$. Since level of consistency is 100%, we conclude that stopping criterion has been satisfied.

There are no more attributes to be discretized and the recursion is now complete. The final set of cut-points are:

height → 8.7, 12.7

weight → 1.6

length → 0.7

And the resulting discretized table is shown in Table 10. However, this table may have excess intervals which should be removed before using the table for rule induction.

Table 10. Discretized table

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8..1.6	0.3..0.7	7.2..8.7	very small
2	0.8..1.6	0.7..2.3	7.2..8.7	small
3	0.8..1.6	0.7..2.3	8.7..12.7	medium
4	0.8..1.6	0.3..0.7	8.7..12.7	medium
5	0.8..1.6	0.7..2.3	8.7..12.7	medium
6	1.6..2.0	0.7..2.3	8.7..12.7	high
7	1.6..2.0	0.7..2.3	12.7..15.2	very high

Post processing. Next we will describe the two-stage merging procedure to address the issue of excessive intervals.

A. Safe merging: For any attribute and for any two neighboring intervals $i..j$ and $j..k$ of the same discretized attribute, if both intervals are labeled by the same decision value, both intervals are merged, *i.e.*, replaced by a new interval $i..k$.

a. ***Weight***: Neighboring intervals

0.8..1.6 → very small

small

medium

1.6..2.0 → high

very high

Since two intervals are differently labeled, they cannot be merged.

b. ***Length***: Neighboring intervals

0.3..0.7 → very small

medium

0.7..2.3 → small

medium

high

very high

Since two intervals are differently labeled, they cannot be merged.

c. ***Height***: Neighboring intervals

7.2..8.7 → very small

small

8.7..12.7 → medium

high

12.7..15.2 → very high

Since all neighboring intervals are differently labeled, they cannot be merged.

B. Proper merging: For any attribute and for any two neighboring intervals $i..j$ and $j..k$ of the same discretized attribute, if a result $i..k$ of merging does not reduce the level of consistency $L(A^D)$, where A^D is the current set of discretized attributes, both intervals are merged (replaced by a new interval *i..k*).

A partition on U constructed from all A-elementary sets of $IND(A)$ is denoted by A^*. For decision variable, $\{d\}$-elementary sets are called concepts, and denoted as $\{d\}^*$. For the discretized table, Table 10:

$\{d\}^* = \{\{1\},\{2\},\{3,4,5\},\{6\},\{7\}\}$

$\{A\}^* = \{\{1\},\{2\},\{3,5\},\{4\},\{6\},\{7\}\}$

Therefore, level of consistency,

$$L(A) = \frac{\left|\underline{A}\{1\}\right| + \left|\underline{A}\{2\}\right| + \left|\underline{A}\{3,4,5\}\right| + \left|\underline{A}\{6\}\right| + \left|\underline{A}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3,4\},\{5\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{1+1+3+1+1}{7}$$

$$= 1$$

a. ***Weight***: After merging 0.8..1.6 and 1.6..2.0

$\{A\}^* = \{\{1\},\{2\},\{3,5,6\},\{4\},\{7\}\}$

And new level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{4\}| + \emptyset + |\{7\}|}{7}$$

$$= \frac{1+1+1+0+1}{7}$$

$$= 0.57$$

Merging intervals lead to reduction in level of consistency. Therefore, they cannot be merged together.

b. ***Length***: After merging 0.3..0.7 and 0.7..2.3

$\{A\}^* = \{\{1,2\},\{3,4,5\},\{6\},\{7\}\}$

And new level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{\emptyset + \emptyset + |\{3,4,5\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{3 + 1 + 1}{7}$$

$$= 0.71$$

Merging intervals lead to reduction in level of consistency. Thus, they also cannot be merged together.

c. ***Height***: *Height* has two cut-points and therefore there are two potential merges for the attribute. First we consider merging neighboring intervals 7.2..8.7 and 8.7..12.7 and then intervals 8.7..12.7 and 12.7..15.2.

- After merging 7.2..8.7 and 8.7..12.7,

  $\{A\}^* = \{\{1,4\},\{2,3,5\},\{6\},\{7\}\}$

  New level of consistency,

  $$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

  $$= \frac{\emptyset + \emptyset + \emptyset + |\{6\}| + |\{7\}|}{7}$$

  $$= \frac{0 + 0 + 0 + 1 + 1}{7}$$

  $$= 0.29$$

  Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged together.

- After merging 8.7..12.7 and 12.7..15.2

  $\{A\}^* = \{\{1\},\{2\},\{3,5\},\{4\},\{6,7\}\}$

  Therefore, new level of consistency,

  $$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3,5\},\{4\}| + \emptyset + \emptyset}{7}$$

$$= \frac{1 + 1 + 3}{7}$$

$$= 0.71$$

Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged either.

Since none of the neighboring intervals could be merged by interval merging, final discretized data set remains same as Table 10.

5.2. Multiple scanning algorithm

We again consider Table 5 to illustrate a worked out example of multiple scanning algorithm.

Scan *t* = 1

A. Find best cut point for each attribute

To evaluate a cut-point q in a variable V, weighted average of class entropies $E(V, q;\ S)$ of the partitioned subsets S_1 and S_2 are determined as:

$$E(V, q; S) = \frac{|S_1|}{|S|} E(S_1) + \frac{|S_2|}{|S|} E(S_2)$$

For each attribute separately, we first sort their unique values and then consider mid-points between adjacent values as potential cut-points. For each potential cut-point, we compute weighted average of class entropy and mark the best cut-point with an asterisk.

Weight: Potential cut-points = 0.8 →**1**← 1.2 →**1.6**← 2.0

Conditional entropy,

$$E(weight, 1;\ U) = 1.536413$$

$E(weight, 1.6, U) = 0.9332607^*$

Length: Potential cut-points = 0.3 →**0.7**← 1.1 →**1.7**← 2.3

Conditional entropy,

$E(length, 0.7, U) = 9.895355$

$E(length, 1.7, U) = 1.536413^*$

Height: Potential cut-points = 7.2 →**8.7**← 10.2 →**12.7**← 15.2

Conditional entropy,

$E(height, 8.7, U) = 1.264965^*$

$E(height, 12.7, U) = 1.536413$

Set of best cut-points:

weight → 1.6

length → 1.7

height → 8.7

Discretized table

Table 11

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8..1.6	0.3..1.7	7.2..8.7	very small
2	0.8..1.6	0.3..1.7	7.2..8.7	small
3	0.8..1.6	0.3..1.7	8.7..15.2	medium
4	0.8..1.6	0.3..1.7	8.7..15.2	medium
5	0.8..1.6	1.7..2.3	8.7..15.2	medium
6	1.6..2.0	1.7..2.3	8.7..15.2	high
7	1.6..2.0	1.7..2.3	8.7..15.2	very high

For the discretized Table 11:

$$\{d\}^* = \{\{1\},\{2\},\{3,4,5\},\{6\},\{7\}\}$$

$$\{A^D\}^* = \{\{1,2\},\{3,4\},\{5\},\{6,7\}\}$$

B. Level of consistency, L(A) is defined as:

$$L(A) = \frac{\sum_{X\in\{d\}^*}|\underline{A}X|}{|U|}$$

Level of consistency is computed as:

$$L(A^D) = \frac{|\underline{A}\{1\}| + |\underline{A}\{2\}| + |\underline{A}\{3,4,5\}| + |\underline{A}\{6\}| + |\underline{A}\{7\}|}{7}$$

$$= \frac{\emptyset + \emptyset + |\{3,4\},\{5\}| + \emptyset + \emptyset}{7}$$

$$= \frac{0+0+3+0+0}{7}$$

$$= 0.43$$

Since $L(A^D) < 1$ and as we can see subsets {1, 2} and {6, 7} in Table 11 are inconsistent, we rescan entire table to distinguish inconsistent subsets shown in Table 12 and Table 13.

Table 12

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8	0.3	7.2	very small
2	0.8	1.1	7.2	small

Table 13

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
6	2.0	2.3	10.2	High
7	2.0	2.3	15.2	very high

Scan $t = 2$

A. Find best cut point for each attribute in Table 12

Weight: Potential cut-points = none

Length: Potential cut-points = 0.3 →**0.7**← 1.1. There is only one possible cut-point.

Height: Potential cut-points = none

Updated set of best cut-points:

weight → 1.6

length → 1.7, 0.7

height → 8.7

Discretized table

Table 14. Discretized table

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8..1.6	0.3..0.7	7.2..8.7	very small
2	0.8..1.6	0.7..1.1	7.2..8.7	small
3	0.8..1.6	0.7..1.1	8.7..15.2	medium
4	0.8..1.6	0.3..0.7	8.7..15.2	medium
5	0.8..1.6	1.7..2.3	8.7..15.2	medium
6	1.6..2.0	1.7..2.3	8.7..15.2	high
7	1.6..2.0	1.7..2.3	8.7..15.2	very high

For the discretized Table 14:

$\{d\}^* = \{\{1\}, \{2\}, \{3, 4, 5\}, \{6\}, \{7\}\}$

$\{A^D\}^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6, 7\}\}$

B. Level of consistency. $L(A^D)$ for Table 14 is computed as:

$$L(A^D) = \frac{\left|\underline{A}\{1\}\right| + \left|\underline{A}\{2\}\right| + \left|\underline{A}\{3,4,5\}\right| + \left|\underline{A}\{6\}\right| + \left|\underline{A}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3\},\{4\},\{5\}| + \emptyset + \emptyset}{7}$$

$$= \frac{1 + 1 + 3 + 0 + 0}{7}$$

$$= 0.714$$

Since $L(A^D) < 1$, rescan entire table to distinguish remaining inconsistent subsets, {6, 7}

Scan *t* = 3

A. Find best cut point for each attribute in Table 13

Weight: Potential cut-points = none

Length: Potential cut-points = none

Height: Potential cut-points = 10.2 →**12.7**← 15.2. There is only one possible cut-point.

Updated set of best cut-points

weight → 1.6

length → 1.7, 0.7

height → 8.7, 12.7

Discretized table

Table 15. Discretization table

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8..1.6	0.3..0.7	7.2..8.7	very small
2	0.8..1.6	0.7..1.7	7.2..8.7	small
3	0.8..1.6	0.7..1.7	8.7..12.7	medium
4	0.8..1.6	0.3..0.7	8.7..12.7	medium
5	0.8..1.6	1.7..2.3	8.7..12.7	medium
6	1.6..2.0	1.7..2.3	8.7..12.7	high
7	1.6..2.0	1.7..2.3	12.7..15.2	very high

For the discretized Table 15:

$\{d\}^* = \{\{1\},\{2\},\{3,4,5\},\{6\},\{7\}\}$

$\{A^D\}^* = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\}$

B. Level of consistency, $L(A^D)$ is defined as:

$$L(A^D) = \frac{\left|\underline{A}\{1\}\right| + \left|\underline{A}\{2\}\right| + \left|\underline{A}\{3,4,5\}\right| + \left|\underline{A}\{6\}\right| + \left|\underline{A}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3\},\{4\},\{5\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{1+1+3+1+1}{7}$$

$$= 1$$

Since $L(A^D) = 1$, we are done.

Post processing

A. **Safe merging:** For any attribute and for any two neighboring intervals $i..j$ and $j..k$ of the same discretized attribute, if both intervals are labeled by the same decision value, both intervals are merged, *i.e.*, replaced by a new interval $i..k$.

a. ***Weight***: Neighboring intervals

0.8..1.6 → very small

small

medium

1.6..2.0 → high

very high

Two intervals are differently labeled. Thus, they cannot be merged.

b. ***Length***: Neighboring intervals

0.3..0.7 → very small

medium

0.7..1.7 → small

medium

1.7..2.3 → medium

high

very high

All intervals are differently labeled. Thus, they cannot be merged.

c. ***Height***: Neighboring intervals

7.2..8.7 → very small

small

8.7..12.7 → medium

high

12.7..15.2 → very high

All neighboring intervals are differently labeled. Thus, they cannot be merged.

B. Proper merging: For any attribute and for any two neighboring intervals $i..j$ and $j..k$ of the same discretized attribute, if a result $i..k$ of merging does not reduce the level of consistency $L(A^D)$, where A^D is the current set of discretized attributes, both intervals are merged (replaced by a new interval $i..k$).

For the discretized Table 15:

$\{d\}^* = \{\{1\},\{2\},\{3,4,5\},\{6\},\{7\}\}$

$\{A\}^* = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\}$

Therefore, level of consistency,

$$L(A) = \frac{\left|\underline{A}\{1\}\right| + \left|\underline{A}\{2\}\right| + \left|\underline{A}\{3,4,5\}\right| + \left|\underline{A}\{6\}\right| + \left|\underline{A}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3\},\{4\},\{5\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{1+1+3+1+1}{7}$$

$$= 1$$

a. ***Weight***: After merging 0.8..1.6 and 1.6..2.0

$$\{A\}^* = \{\{1\},\{2\},\{3\},\{4\},\{5,6\},\{7\}\}$$

And level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3\},\{4\}| + \emptyset + |\{7\}|}{7}$$

$$= \frac{1+1+2+0+1}{7}$$

$$= 0.714$$

Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged.

b. ***Length***: After merging 0.3..0.7 and 0.7..1.7

$$\{A\}^* = \{\{1,2\},\{3,4\},\{5\},\{6\},\{7\}\}$$

Therefore, new level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{\emptyset + \emptyset + |\{3,4\},\{5\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{3+1+1}{7}$$

$$= 0.714$$

Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged.

After merging 0.7..1.7 and 1.7..2.3

$$\{A\}^* = \{\{1\},\{2\},\{3,5\},\{4\},\{6\},\{7\}\}$$

Level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3,5\},\{4\}| + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{1+1+3+1+1}{7}$$

$$= 1$$

Merging of intervals did not lead to reduction in level of consistency. Thus, 0.7..1.7 and 1.7..2.3 can be merged together as 0.7..2.3. Discretization is updated in Table 16.

Table 16

CASE	ATTRIBUTE			DECISION
	weight	length	height	price
1	0.8..1.6	0.3..0.7	7.2..8.7	very small
2	0.8..1.6	0.7..2.3	7.2..8.7	small
3	0.8..1.6	0.7..2.3	8.7..12.7	medium
4	0.8..1.6	0.3..0.7	8.7..12.7	medium
5	0.8..1.6	0.7..2.3	8.7..12.7	medium
6	1.6..2.0	0.7..2.3	8.7..12.7	high
7	1.6..2.0	0.7..2.3	12.7..15.2	very high

c. ***Height***:

After merging 7.2..8.7 and 8.7..12.7

$$\{A\}^* = \{\{1,4\},\{2,3,5\},\{6\},\{7\}\}$$

Therefore, new level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{\emptyset + \emptyset + \emptyset + |\{6\}| + |\{7\}|}{7}$$

$$= \frac{0 + 0 + 0 + 1 + 1}{7}$$

$$= 0.29$$

Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged.

After merging 8.7..12.7 and 12.7..15.2

$$\{A\}^* = \{\{1\},\{2\},\{3,5\},\{4\},\{6,7\}\}$$

Therefore, new level of consistency,

$$L(B) = \frac{\left|\underline{B}\{1\}\right| + \left|\underline{B}\{2\}\right| + \left|\underline{B}\{3,4,5\}\right| + \left|\underline{B}\{6\}\right| + \left|\underline{B}\{7\}\right|}{7}$$

$$= \frac{|\{1\}| + |\{2\}| + |\{3,5\},\{4\}| + \emptyset + \emptyset}{7}$$

$$= \frac{1 + 1 + 3}{7}$$

$$= 0.714$$

Merging intervals lead to reduction in level of consistency. Thus, they cannot be merged.

CHAPTER 6. RESULTS AND DISCUSSION

6.1. Discretization results

Summary of discretization by dominant attribute and multiple scanning approach is shown in Table 17 - Table 33. Dominant attribute approach is shown with scan count, $t = 0$ whereas all other experiments are conducted by using multiple scanning approach with respective scan counts as shown. In general, multiple scanning approach is more conservative than dominant attribute approach which is apparent from consistently fewer number of intervals produced by the multiple scanning approach. Further, results indicate that rule sets produced by multiple scanning approach are more compact i.e., total number of rules and conditions produced is lower and the proportion of conditions per rule is higher. After few scans, variations with respect to number of intervals stabilized and this stabilization was more prominent post-processing step of interval merging was completed. For example, data sets *m-bank*, *m-echo*, *m-globe*, *m-image*, *m-iris*, *m-wine*, *price*, *table* and *trip* had no variation from scan numbers 1, 6, 5, 7, 6, 1, 1, 1 and 1 respectively (Table 52).

Table 17. Summary of discretization for austr

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	162	11.57	48	3.43
1	54	3.86	36	2.57
2	60	4.29	35	2.5
3	65	4.64	34	2.43
4	69	4.93	35	2.5
5	74	5.29	35	2.5
6	79	5.64	35	2.5
7	83	5.93	35	2.5
8	86	6.14	37	2.64
9	90	6.43	37	2.64
10	97	6.93	36	2.57

Table 18. Summary of discretization for common_combined_lers

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	16326	1	16306	1
1	32523	2	16293	1

Table 19. Summary of discretization for m-bank

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	14	2.8	14	2.8
1	15	3	8	1.6
2	15	3	8	1.6
3	15	3	8	1.6
4	15	3	8	1.6
5	15	3	8	1.6
6	15	3	8	1.6
7	15	3	8	1.6
8	15	3	8	1.6
9	15	3	8	1.6
10	15	3	8	1.6

Table 20. Summary of discretization for m-echo

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	32	4.57	21	3
1	26	3.71	17	2.43
2	31	4.43	19	2.71
3	35	5	20	2.86
4	39	5.57	20	2.86
5	42	6	21	3
6	46	6.57	21	3
7	46	6.57	21	3
8	46	6.57	21	3
9	46	6.57	21	3
10	46	6.57	21	3

Table 21. Summary of discretization for m-glass

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	100	11.11	44	4.89
1	54	6	29	3.22
2	66	7.33	35	3.89
3	72	8	34	3.78
4	77	8.56	33	3.67
5	81	9	32	3.56
6	79	8.78	32	3.56
7	89	9.89	32	3.56
8	107	11.89	32	3.56
9	112	12.44	32	3.56
10	117	13	33	3.67

Table 22. Summary of discretization for m-globe

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	25	5	19	3.8
1	30	6	16	3.2
2	26	5.2	15	3
3	33	6.6	16	3.2
4	39	7.8	16	3.2
5	42	8.4	16	3.2
6	42	8.4	16	3.2
7	42	8.4	16	3.2
8	42	8.4	16	3.2
9	42	8.4	16	3.2
10	42	8.4	16	3.2

Table 23. Summary of discretization for m-image

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	86	4.53	47	2.47
1	63	3.32	38	2
2	78	4.11	33	1.74
3	91	4.79	36	1.89
4	106	5.58	40	2.11
5	120	6.32	39	2.05
6	131	6.89	41	2.16
7	140	7.37	43	2.26
8	140	7.37	43	2.26
9	140	7.37	43	2.26
10	140	7.37	43	2.26

Table 24. Summary of discretization for m-iris

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	15	3.75	11	2.75
1	21	5.25	11	2.75
2	21	5.25	10	2.5
3	23	5.75	11	2.75
4	25	6.25	11	2.75
5	27	6.75	11	2.75
6	28	7	11	2.75
7	28	7	11	2.75
8	28	7	11	2.75
9	28	7	11	2.75
10	28	7	11	2.75

Table 25. Summary of discretization for m-wine

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	29	2.23	25	1.92
1	26	2	21	1.62
2	26	2	21	1.62
3	26	2	21	1.62
4	26	2	21	1.62
5	26	2	21	1.62
6	26	2	21	1.62
7	26	2	21	1.62
8	26	2	21	1.62
9	26	2	21	1.62
10	26	2	21	1.62

Table 26. Summary of discretization for n-abalone

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	1166	145.75	319	39.88
1	1130	141.25	320	40
2	1129	141.13	324	40.5
3	1144	143	321	40.13
4	1147	143.38	317	39.63
5	1169	146.13	316	39.5
6	1177	147.13	309	38.63
7	1198	149.75	312	39
8	1217	152.13	310	38.75
9	1238	154.75	314	39.25
10	1252	156.5	318	39.75

Table 27. Summary of discretization for n-bupa

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	83	13.83	35	5.83
1	86	14.33	32	5.33
2	87	14.5	34	5.67
3	90	15	38	6.33
4	88	14.67	38	6.33
5	88	14.67	38	6.33
6	91	15.17	34	5.67
7	95	15.83	35	5.83
8	101	16.83	36	6
9	106	17.67	38	6.33
10	107	17.83	36	6

Table 28. Summary of discretization for n-eoli

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	65	9.29	34	4.86
1	58	8.29	34	4.86
2	64	9.14	33	4.71
3	72	10.29	32	4.57
4	78	11.14	34	4.86
5	80	11.43	34	4.86
6	85	12.14	34	4.86
7	90	12.86	35	5
8	97	13.86	36	5.14
9	99	14.14	36	5.14
10	106	15.14	36	5.14

Table 29. Summary of discretization for n-pima

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	148	18.5	41	5.13
1	108	13.5	42	5.25
2	112	14	41	5.13
3	118	14.75	42	5.25
4	121	15.13	43	5.38
5	126	15.75	40	5
6	129	16.13	43	5.38
7	132	16.5	43	5.38
8	143	17.88	44	5.5
9	148	18.5	44	5.5
10	154	19.25	45	5.63

Table 30. Summary of discretization for n-wave-512

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	174	8.29	59	2.81
1	85	4.05	45	2.14
2	105	5	43	2.05
3	124	5.9	43	2.05
4	142	6.76	43	2.05
5	161	7.67	43	2.05
6	177	8.43	45	2.14
7	197	9.38	46	2.19
8	212	10.1	48	2.29
9	231	11	47	2.24
10	250	11.9	47	2.24

Table 31. Summary of discretization for price

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	7	2.33	7	2.33
1	8	2.67	7	2.33
2	8	2.67	7	2.33
3	8	2.67	7	2.33
4	8	2.67	7	2.33
5	8	2.67	7	2.33
6	8	2.67	7	2.33
7	8	2.67	7	2.33
8	8	2.67	7	2.33
9	8	2.67	7	2.33
10	8	2.67	7	2.33

Table 32. Summary of discretization for table

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	7	2.33	7	2.33
1	8	2.67	7	2.33
2	8	2.67	7	2.33
3	8	2.67	7	2.33
4	8	2.67	7	2.33
5	8	2.67	7	2.33
6	8	2.67	7	2.33
7	8	2.67	7	2.33
8	8	2.67	7	2.33
9	8	2.67	7	2.33
10	8	2.67	7	2.33

Table 33. Summary of discretization for trip

Scans	Before interval merging		After interval merging	
	# intervals	# intervals/attribute	# intervals	# intervals/attribute
0	9	3	8	2.67
1	10	3.33	8	2.67
2	10	3.33	8	2.67
3	10	3.33	8	2.67
4	10	3.33	8	2.67
5	10	3.33	8	2.67
6	10	3.33	8	2.67
7	10	3.33	8	2.67
8	10	3.33	8	2.67
9	10	3.33	8	2.67
10	10	3.33	8	2.67

Table 34 shows percent reduction in interval counts after merging operations. Results indicate that, on average, multiple scanning approach produces comparatively excessive intervals which is associated with high reduction rate during merging process. Although both discretization approaches are global, dominant attribute approach is less global in a sense that it focuses on only one attribute (dominant) and selects one cut-point on every iteration for splitting data set. On the other hand, multiple scanning approach selects as many cut-points as the number of attributes in each iteration and hence a more global approach.

Table 34.Percent reduction in interval counts after preprocessing

Data	**% Reduction after interval merging**	
	Dominant attribute approach	**Multiple Scanning Approach (Average)**
austr	70.37	51.70
common_combined_lers	0.12	49.90
m-bank	0.00	46.67
m-echo	34.38	48.66
m-glass	56.00	60.05
m-globe	24.00	57.09
m-image	45.35	63.34
m-iris	26.67	57.10
m-wine	13.79	19.23
n-abalone	72.64	73.17
n-bupa	57.83	61.58
n-ecoli	47.69	57.30
n-pima	72. 30	66.60
n-wave-512	66.09	70.39
price	0.00	12.50
table	0.00	12.50
trip	11.11	20.00

6.2. Discretization results of bankruptcy data

6.2.1. Dominant attribute approach

The approach tends to confine discriminating features to dominant attributes only. Figure 4 - Figure 8 shows that the algorithm has repeatedly found attribute a_1 as most informative and therefore this attribute was overly discretized into large number of intervals. Overall, there were 14 intervals defined for the entire data set but as many as 9 intervals were confined to dominant attributes, a_1. It took 14 rules to explain all patterns in the data set.

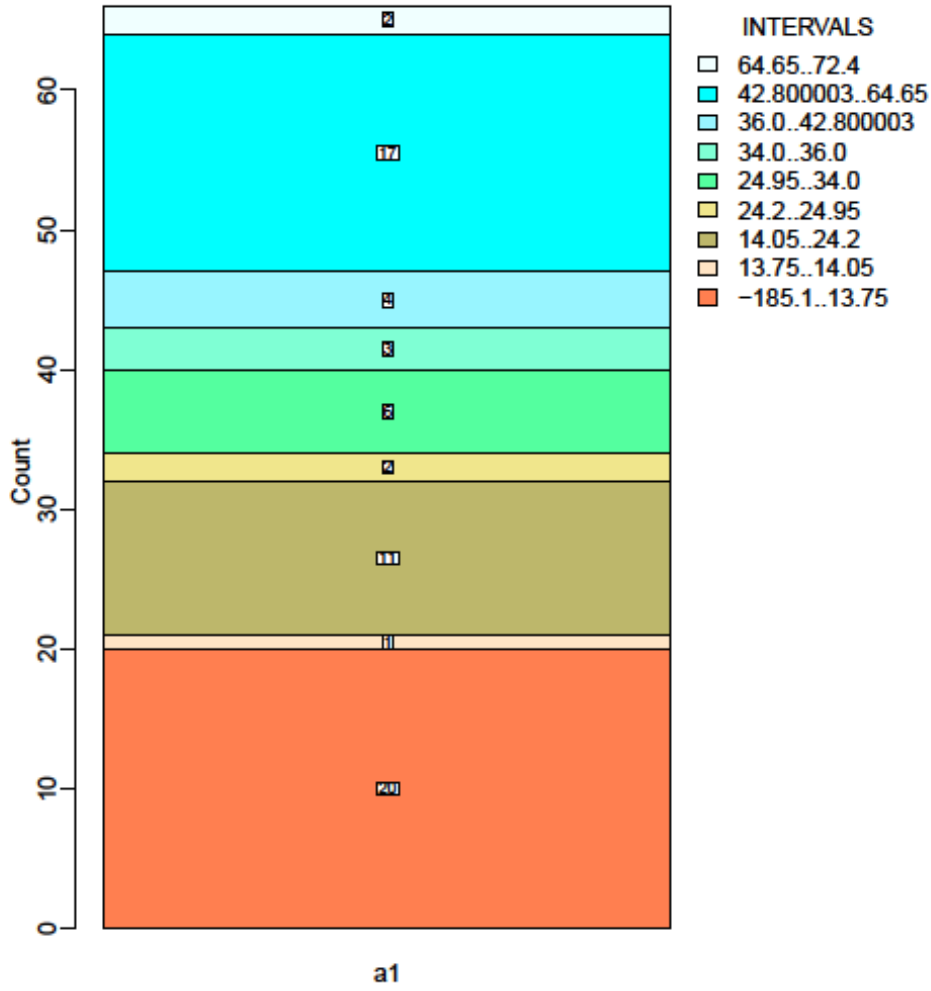

Figure 4. DM: Interval distribution for attribute, a1

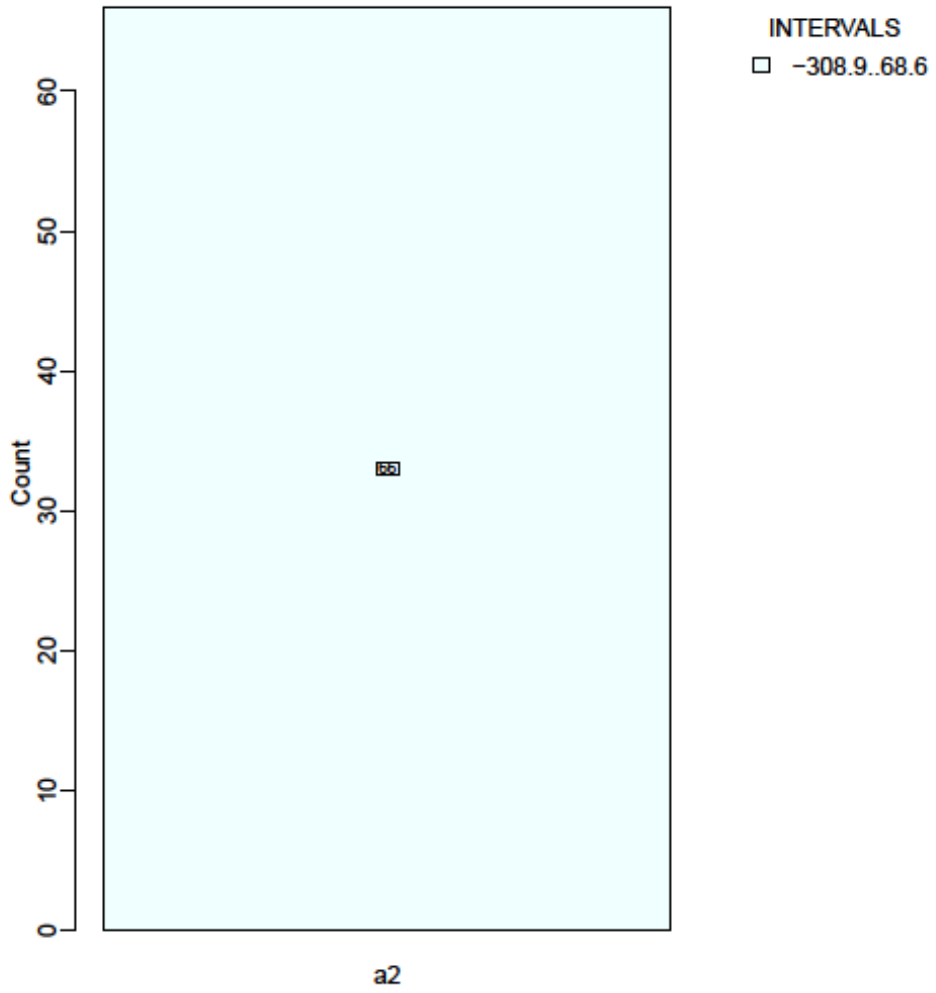

Figure 5. DM: Interval distribution for attribute, a2

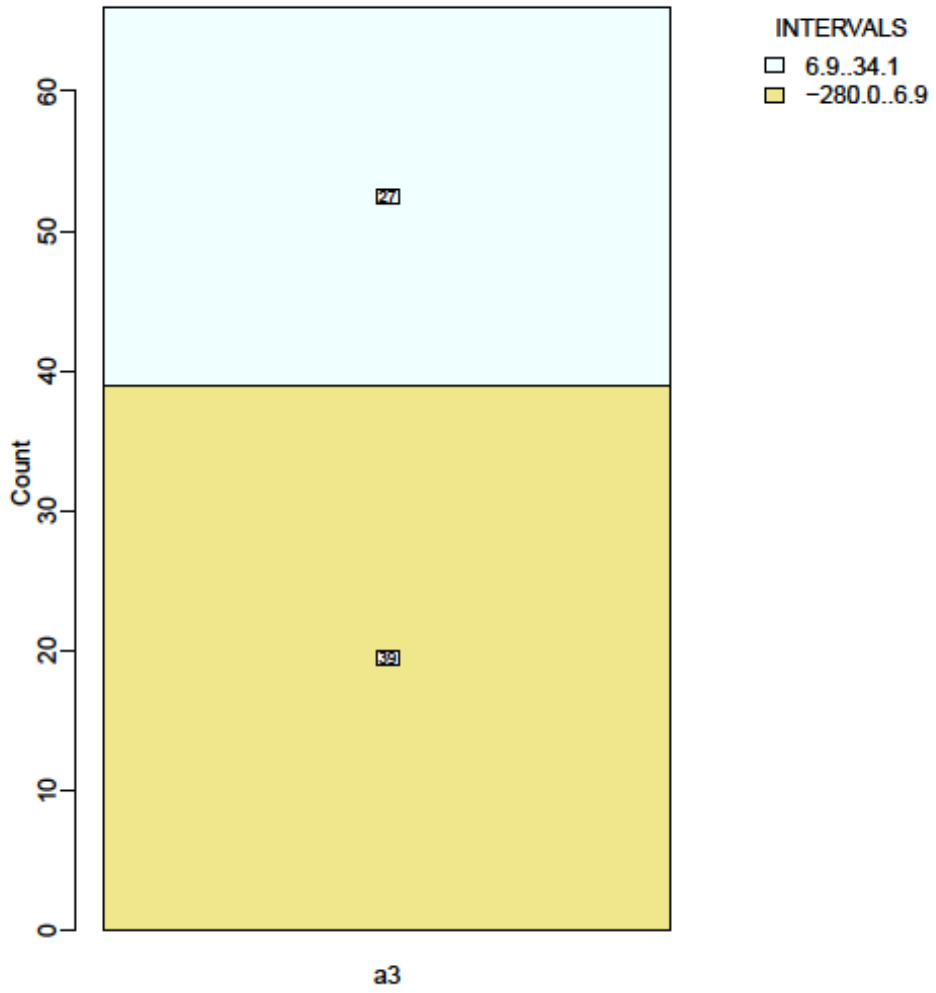

Figure 6. DM: Interval distribution for attribute, a3

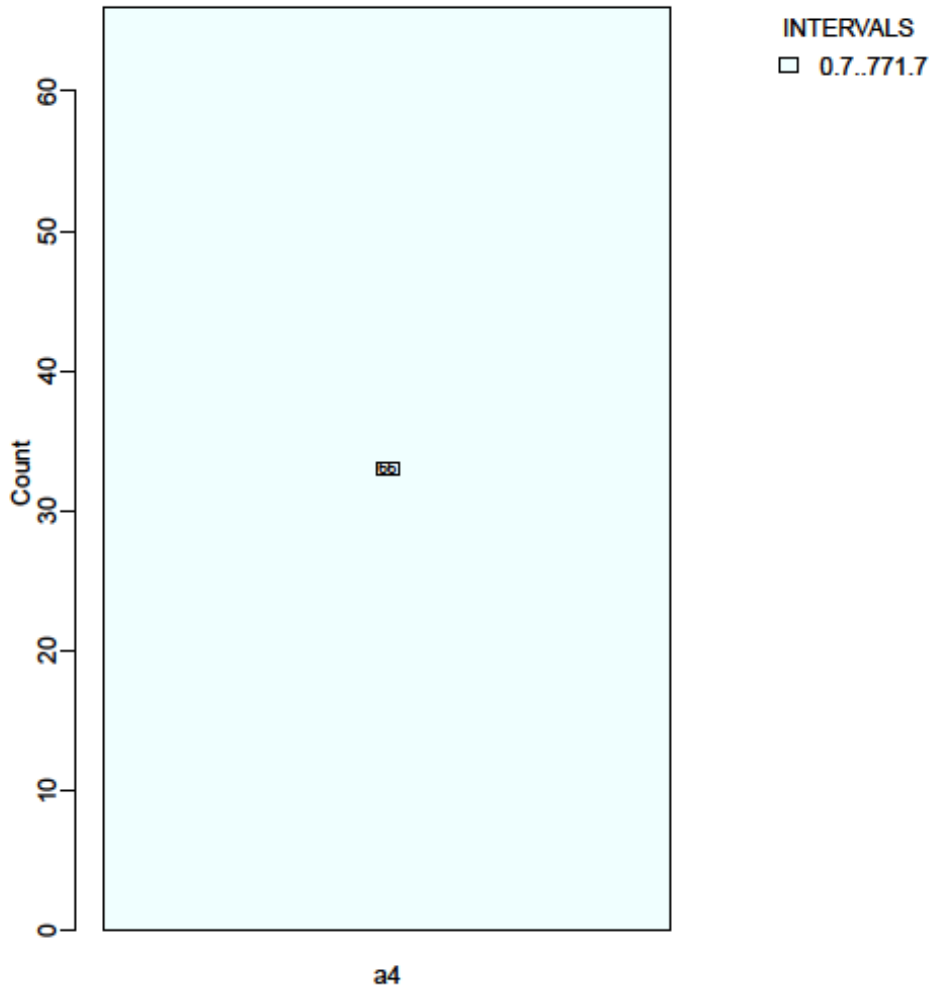

Figure 7. DM: Interval distribution for attribute, a4

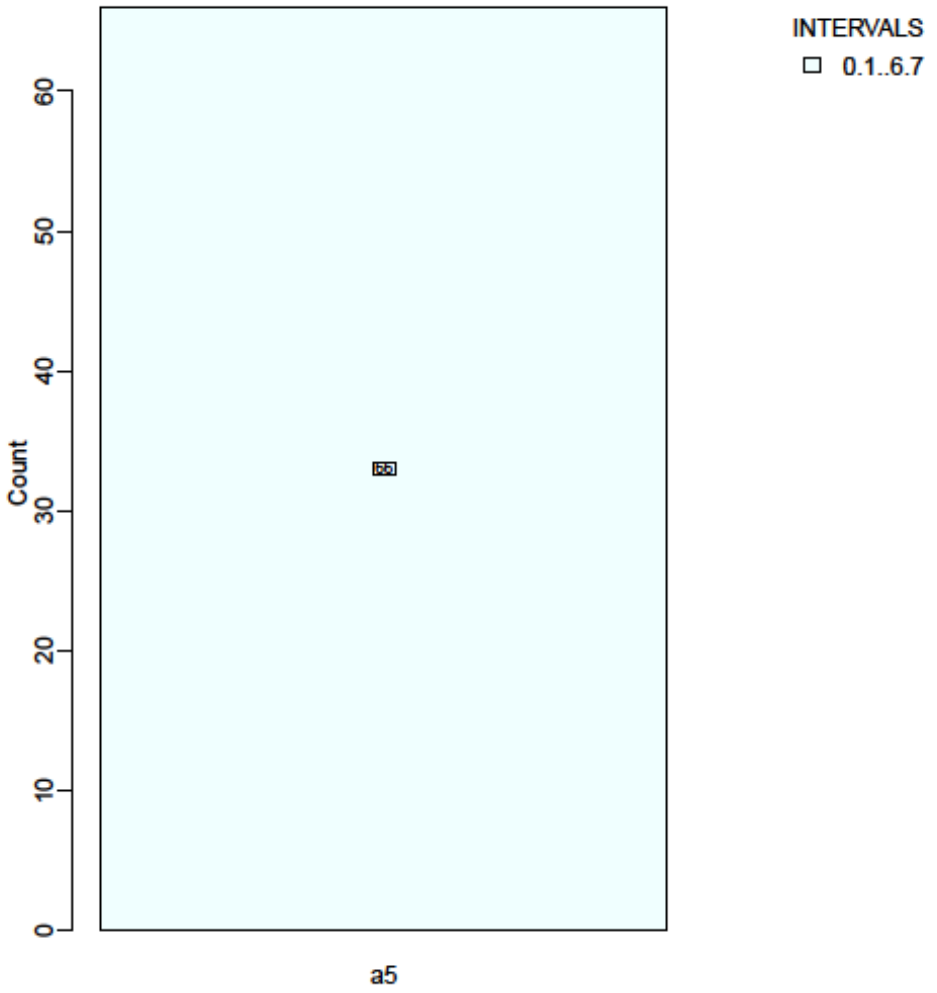

Figure 8. DM: Interval distribution for attribute, a5

6.2.2. Multiple scanning approach (10 scans)

This approach tends to distribute discriminating features evenly across all attributes and explains patterns with comparatively much less number of rules. Results in Figure 9 - Figure 13 show that the approach has discretized dataset into 8 intervals, 6 of them are discriminating and spread across 3 attributes. Entire data set was explained with just 4 rules.

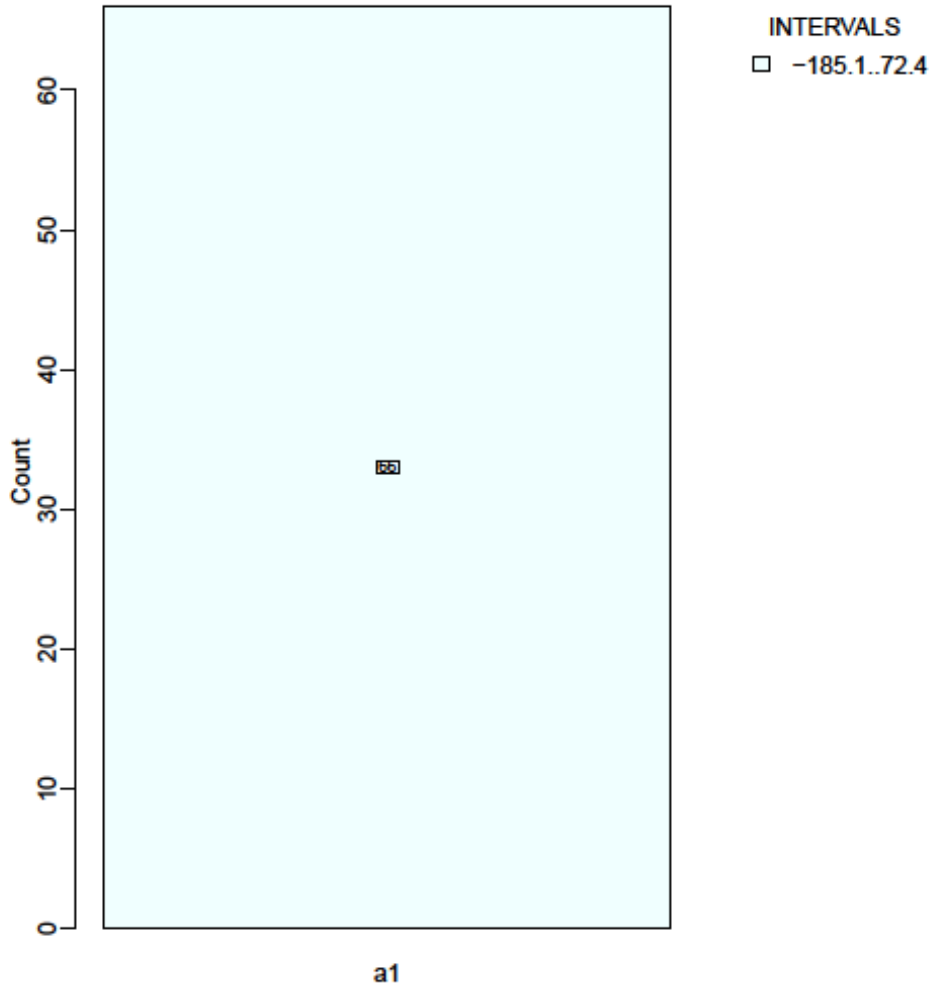

Figure 9. MS: Interval distribution for attribute, a1

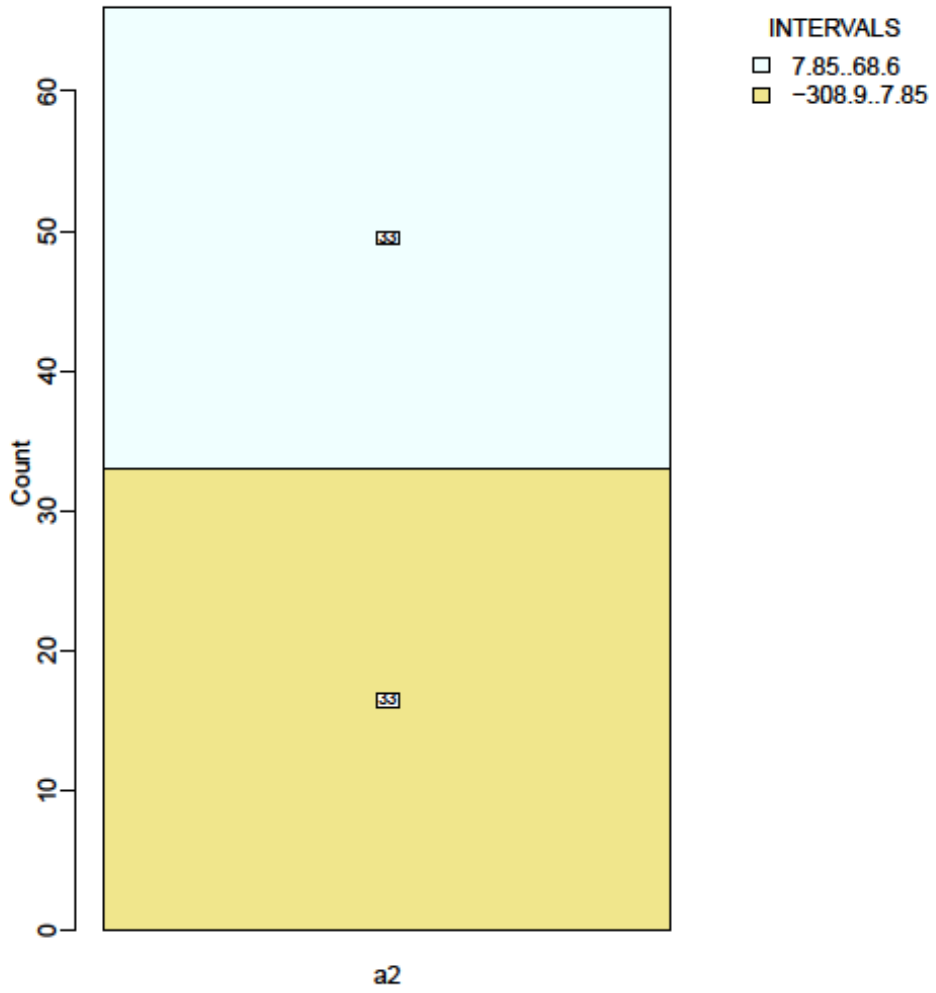

Figure 10. MS: Interval distribution for attribute, a2

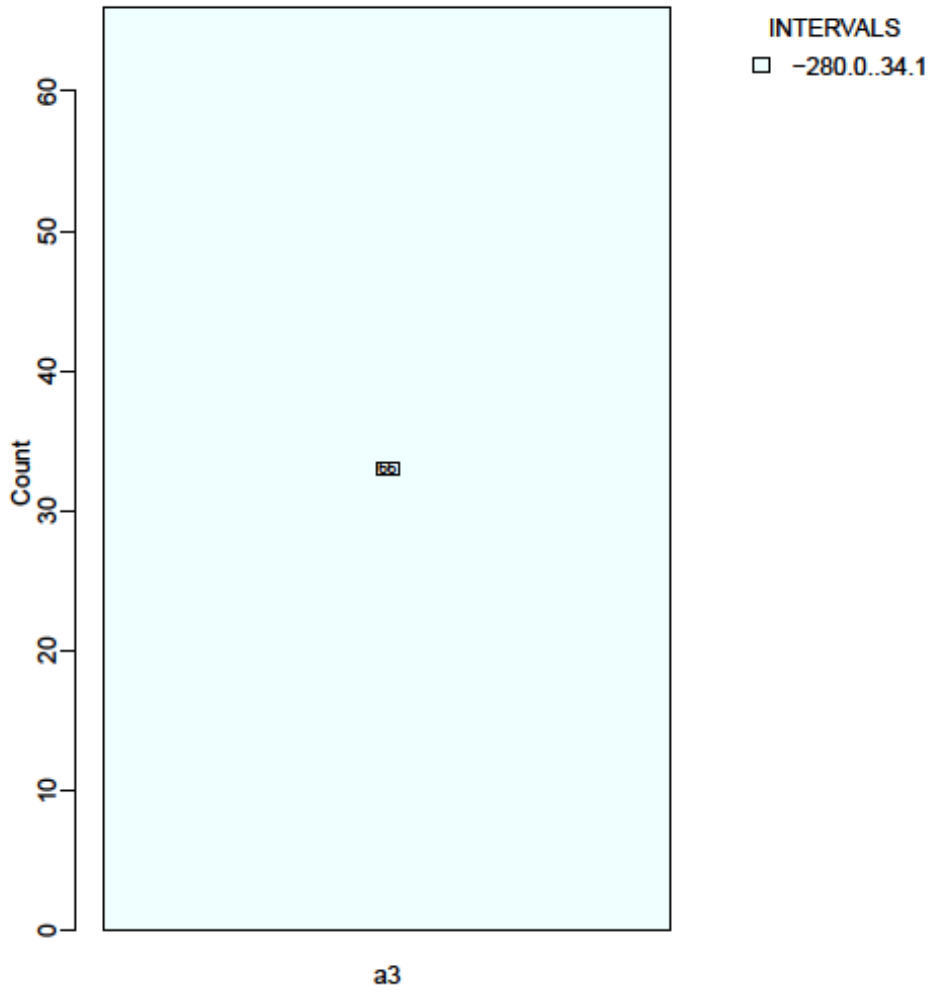

Figure 11. MS: Interval distribution for attribute, a3

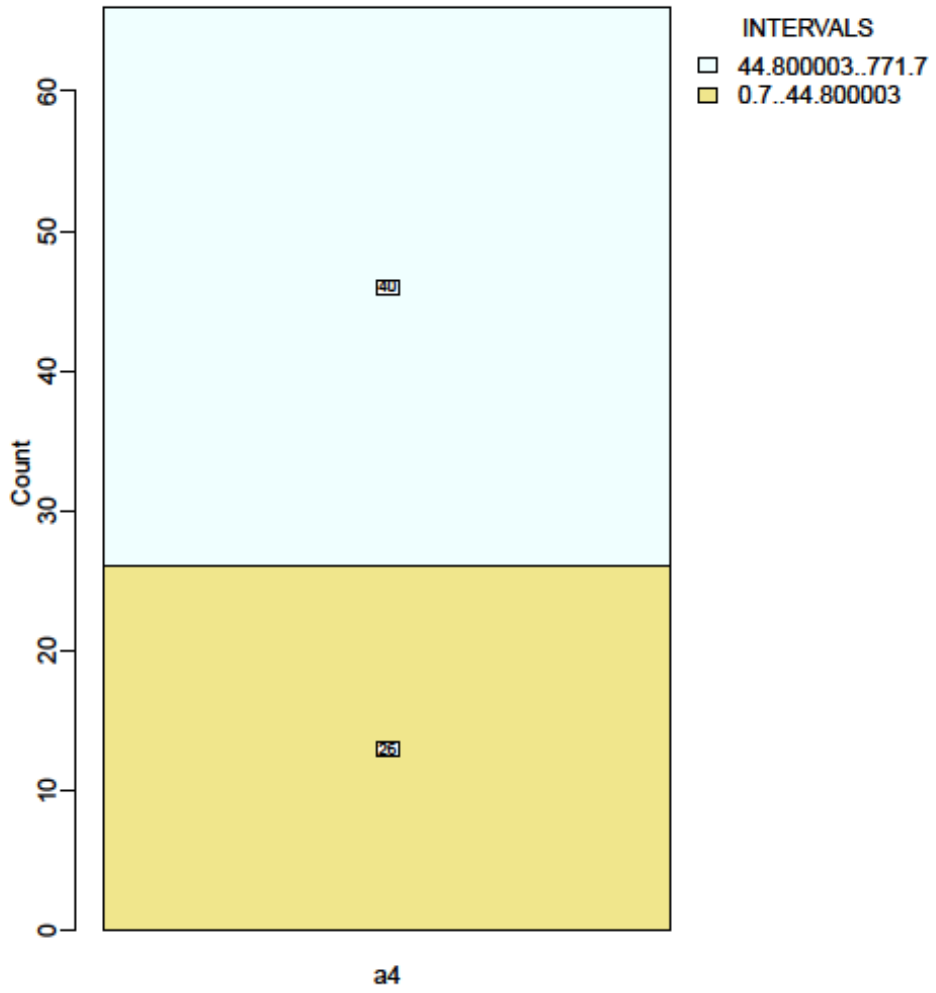

Figure 12. MS: Interval distribution for attribute, a4

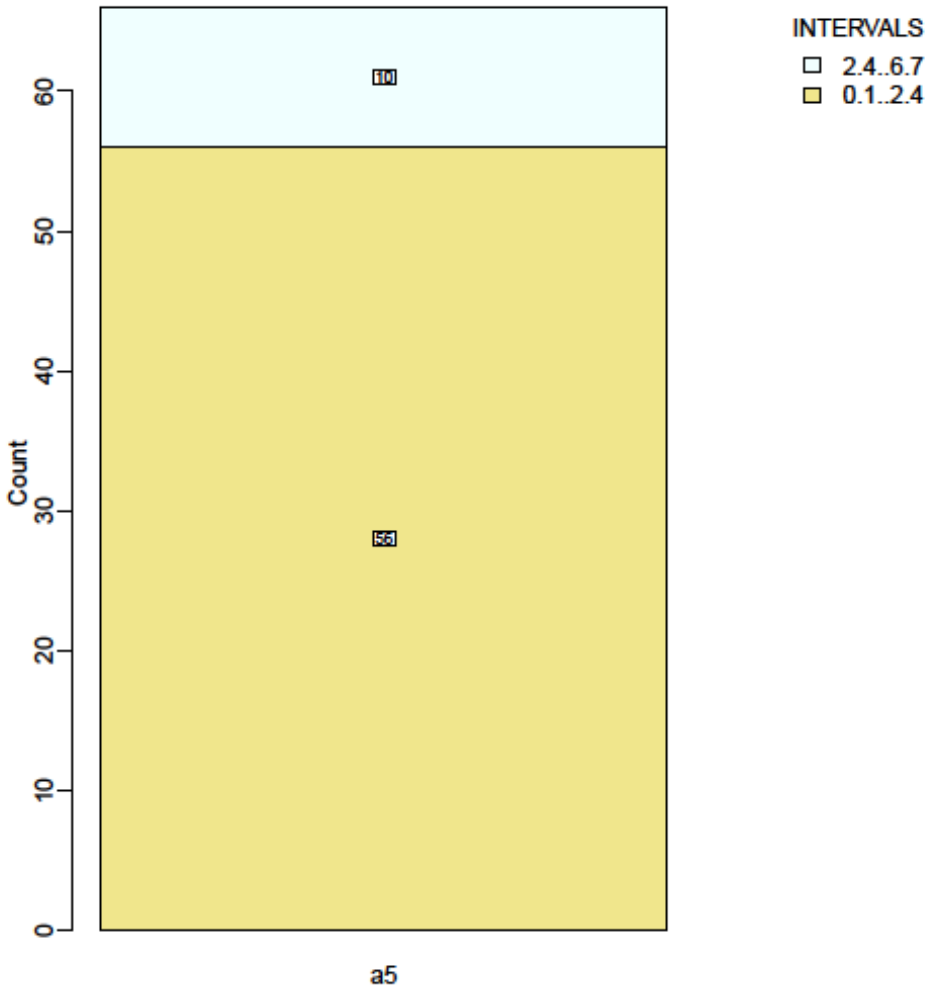

Figure 13. MS: Interval distribution for attribute, a5

6.3. LEM2 induced rules

Table 35 - Table 51 shows the general trend that the rule set derived from data set discretized by dominant attribute approach contains more number of rules and conditions as compared to multiple scanning approach.

Table 35. LEM2 induced rules for austr

Scans	Rules	Conditions	Conditions/rule
0	174	563	3.2356
1	99	489	4.9394
2	116	539	4.6466
3	127	561	4.4173
4	123	577	4.6911
5	124	581	4.6855
6	121	544	4.4959
7	127	570	4.4882
8	126	542	4.3016
9	125	528	4.224
10	125	535	4.28

Table 36. LEM2 induced rules for common_combined_lers

Scans	Rules	Conditions	Conditions/Rule
0	67	67	1
1	67	67	1

Table 37. LEM2 induced rules for m-bank

Scans	Rules	Conditions	Conditions/Rule
0	10	14	1.4
1	4	7	1.75
2	4	7	1.75
3	4	7	1.75
4	4	7	1.75
5	4	7	1.75
6	4	7	1.75
7	4	7	1.75
8	4	7	1.75
9	4	7	1.75
10	4	7	1.75

Table 38. LEM2 induced rules for m-echo

Scans	Rules	Conditions	Conditions/rule
0	31	72	2.3226
1	24	64	2.6667
2	23	65	2.8261
3	29	80	2.7586
4	28	77	2.75
5	25	68	2.72
6	25	68	2.72
7	25	68	2.72
8	25	68	2.72
9	25	68	2.72
10	25	68	2.72

Table 39. LEM2 induced rules for m-glass

Scans	Rules	Conditions	Conditions/rule
0	98	225	2.2959
1	64	223	3.4844
2	75	252	3.36
3	80	271	3.3875
4	76	264	3.4737
5	77	271	3.5195
6	70	238	3.4
7	69	244	3.5362
8	81	278	3.4321
9	80	274	3.425
10	83	268	3.2289

Table 40. LEM2 induced rules for m-globe

Scans	Rules	Conditions	Conditions/rule
0	27	57	2.1111
1	24	57	2.375
2	22	56	2.5455
3	22	53	2.4091
4	20	46	2.3
5	20	46	2.3
6	20	46	2.3
7	20	46	2.3
8	20	46	2.3
9	20	46	2.3
10	20	46	2.3

Table 41. LEM2 induced rules for m-image

Scans	Rules	Conditions	Conditions/rule
0	57	64	1.1228
1	64	87	1.3594
2	51	59	1.1569
3	55	71	1.2909
4	62	77	1.2419
5	59	67	1.1356
6	57	65	1.1404
7	58	64	1.1034
8	58	64	1.1034
9	58	64	1.1034
10	58	64	1.1034

Table 42. LEM2 induced rules for m-iris

Scans	Rules	Conditions	Conditions/rule
0	10	23	2.3
1	11	23	2.0909
2	10	21	2.1
3	10	22	2.2
4	10	22	2.2
5	10	22	2.2
6	10	20	2
7	10	20	2
8	10	20	2
9	10	20	2
10	10	20	2

Table 43. LEM2 induced rules for m-wine

Scans	Rules	Conditions	Conditions/rule
0	24	57	2.375
1	11	37	3.3636
2	11	37	3.3636
3	11	37	3.3636
4	11	37	3.3636
5	11	37	3.3636
6	11	37	3.3636
7	11	37	3.3636
8	11	37	3.3636
9	11	37	3.3636
10	11	37	3.3636

Table 44. LEM2 induced rules for n-abalone

Scans	Rules	Conditions	Conditions/rule
0	3135	10624	3.3888
1	3161	10551	3.3379
2	3184	10607	3.3313
3	3178	10593	3.3332
4	3163	10589	3.3478
5	3146	10527	3.3462
6	3152	10574	3.3547
7	3157	10537	3.3377
8	3147	10522	3.3435
9	3161	10539	3.3341
10	3155	10508	3.3306

Table 45. LEM2 induced rules for n-bupa

Scans	Rules	Conditions	Conditions/rule
0	154	465	3.0195
1	11	37	3.3636
2	150	476	3.1733
3	159	479	3.0126
4	145	435	3
5	154	455	2.9545
6	143	433	3.028
7	153	453	2.9608
8	151	453	3
9	162	485	2.9938
10	146	443	3.0342

Table 46. LEM2 induced rules for n-ecoli

Scans	Rules	Conditions	Conditions/rule
0	99	266	2.6869
1	103	284	2.7573
2	113	322	2.8496
3	111	319	2.8739
4	113	320	2.8319
5	111	308	2.7748
6	116	316	2.7241
7	108	299	2.7685
8	114	329	2.886
9	113	319	2.823
10	114	319	2.7982

Table 47. LEM2 induced rules for b-pima

Scans	Rules	Conditions	Conditions/rule
0	285	958	3.3614
1	256	952	3.7188
2	263	933	3.5475
3	272	988	3.6324
4	275	993	3.6109
5	256	930	3.6328
6	256	904	3.5312
7	270	965	3.5741
8	263	939	3.5703
9	263	939	3.5703
10	264	932	3.5303

Table 48. LEM2 induced rules for n-wave-512

Scans	Rules	Conditions	Conditions/rule
0	182	667	3.6648
1	104	559	5.375
2	105	510	4.8571
3	107	512	4.785
4	129	549	4.2558
5	120	525	4.375
6	118	526	4.4576
7	179	771	4.3073
8	172	746	4.3372
9	169	722	4.2722
10	171	768	4.4912

Table 49. LEM2 induced rules for price

Scans	Rules	Conditions	Conditions/rule
0	5	9	1.8
1	5	9	1.8
2	5	9	1.8
3	5	9	1.8
4	5	9	1.8
5	5	9	1.8
6	5	9	1.8
7	5	9	1.8
8	5	9	1.8
9	5	9	1.8
10	5	9	1.8

Table 50. LEM2 induced rules for table

Scans	Rules	Conditions	Conditions/rule
0	5	9	1.8
1	5	9	1.8
2	5	9	1.8
3	5	9	1.8
4	5	9	1.8
5	5	9	1.8
6	5	9	1.8
7	5	9	1.8
8	5	9	1.8
9	5	9	1.8
10	5	9	1.8

Table 51. LEM2 induced rules for trip

Scans	Rules	Conditions	Conditions/rule
0	8	15	1.875
1	7	12	1.7143
2	7	12	1.7143
3	7	12	1.7143
4	7	12	1.7143
5	7	12	1.7143
6	7	12	1.7143
7	7	12	1.7143
8	7	12	1.7143
9	7	12	1.7143
10	7	12	1.7143

CHAPTER 7. CONCLUSIONS

Preliminary results presented in this study are consistent with earlier studies [5, 6], which indicate that multiple scanning approach performs better than dominant attribute approach by producing comparatively smaller and simpler rule sets. It was consistently observed from the results of multiple scanning approach that after scanning dataset for few iterations, variations with respect to number of intervals produced dampened significantly. Table 52 shows that after certain number of scans, further scanning did not affect outcome with respect to the number of intervals produced and the number of rules induced.

Table 52. Variation dampening effect with MSA

Data	# scans	# intervals before merging	# intervals after merging	LEM2 rules	
				# rules	# conditions
m-bank	1 - 10	15	8	4	7
m-echo	6 - 10	46	21	25	68
m-globe	5 - 10	42	16	20	46
m-image	7 - 10	140	43	58	64
m-iris	6 - 10	28	11	10	20
m-wine	1 - 10	26	21	11	37
price	1 - 10	8	7	5	9
table	1 - 10	8	7	5	9
trip	1 - 10	10	8	7	12

Further, number of intervals after merging operation consistently showed greater stabilization than those before merging operation. However, claim should be validated with more elaborate experiments and statistical tests. The two approaches presented here affirms the promise of entropy based approaches in discretization which clearly has a scope for further improvement. Besides incorporating novel ideas such as integrating discretization with merging procedures, immediate improvements can be achieved by implementing more efficient algorithms. The current implementation works well for moderately sized dataset but failed to produce result in a reasonable amount of time for bigger sized data set. Expense of time complexity becomes

obvious with increasing input size and as an example, among the studied datasets, *common_combined_lers* with 68 cases and 16280 attributes was too prohibitive to permit us from repeating experiments beyond one scan. The complex and recursive nature of algorithm can be restrictive but not prohibitive towards better implementation. We are hoping to ameliorated cost by implementing efficient program with more sophisticated data-structures and switching to platform-dependent programming language such as C++.

APPENDICES

Following source files are included as an attachments

Java source code

- *DomAttrApp.java* - Dominant attribute approach.
- *MultScanApp.java* - Multiple scanning approach.

R source code

- *rough-set.R* - Figure 1. Rough sets.
- *barplot.R* - Figure 2. Probability distribution.
- *interval-count-da.R* - Figure 4 - Figure 8. Interval distributions with dominant attribute approach.
- *interval-count-ms.R* - Figure 9 - Figure 13. Interval distributions with multiple scanning approach

REFERENCES

1. Russell, S.J. and P. Norvig, *Artificial intelligence : a modern approach*. 3rd ed. Prentice Hall series in artificial intelligence. 2010, Upper Saddle River, N.J.: Prentice Hall. xviii, 1132 p.

2. Carbonell, J.G., R.S. Michalski, and T.M. Mitchell, *An overview of machine learning*, in *Machine learning : an artificial intelligence approach*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Editors. 1983, Morgan Kaufmann: Los Altos, Calif. p. 3-23.

3. Witten, I.H., E. Frank, and M.A. Hall, *Preface*, in *Data mining : practical machine learning tools and techniques*. 2011, Morgan Kaufmann: Burlington, MA. p. xxi-xxiv.

4. Grzymala-Busse, J.W., *Introduction*, in *Managing uncertainty in expert systems*. 1991, Kluwer Academic: Boston. p. xix, 224 p.

5. Grzymala-Busse, J.W. *A multiple scanning strategy for entropy based discretization*. in *Proceedings of the 18-th International Symposium on Methodologies for Intelligent Systems, ISMIS 2009*. 2009. Prague, Czech Republic: Springer-Verlag Berlin Heidelberg 2009.

6. Grzymala-Busse, J.W., *Discretization based on entropy and multiple scanning.* Entropy, 2013. **15**(5): p. 1486-1502.

7. Weisberg, S., *Scatterplots and Regression*, in *Applied linear regression*. 2014, Wiley-Blackwell. p. 1.

8. Maimon, O. and L. Rokach, *Introduction to Knowledge Discovery and Data Mining*, in *Data mining and knowledge discovery handbook*, O. Maimon and L. Rokach, Editors. 2010, Springer: Boston, MA. p. 1-15.

9. Pawlak, Z., *Rough sets.* International Journal of Computer & Information Sciences, 1982. **11**(5): p. 341-356.

10. Grzymala-Busse, J.W., *Rule Induction*, in *Data mining and knowledge discovery handbook*, O. Maimon and L. Rokach, Editors. 2010, Springer. p. 249-265.

11. Grzymala-Busse, J., *LERS—A Data Mining System*, in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Editors. 2005, Springer US. p. 1347-1351.

12. Grzymala-Busse, J.W., *Rule induction from rough approximations.*

13. Grzymala-Busse, J.W., *A new version of the rule induction system LERS.* Fundam. Inf., 1997. **31**(1): p. 27-39.

14. Ott, L. and M. Longnecker, *Probability and probability distributions*, in *A first course in statistical methods*. 2004, Thomson-Brooks/Cole: Belmont, CA. p. 109-172.

15. Cover, T.M. and J.A. Thomas, *Introduction and preview*, in *Elements of information theory*. 2006, Wiley-Interscience: Hoboken, N.J. p. 1-12.

16. Shannon, C.E., *A mathematical theory of communication.* The Bell System Technical Journal, 1948. **27**(3): p. 379–423.

17. Blajdo, P., et al. *A comparison of six approaches to discretization - A rough set perspective*. in *Proceedings of the Rough Sets and Knowledge Technology, RSKT'2008 Conference*. 2008. Chengdu, China: Springer Verlag.

18. Chmielewski, M.R. and J.W. Grzymala-Busse. *Global discretization of continuous attributes as preprocessing for machine learning*. in *Proceedings of the 3-rd International Workshop on Rough Sets and Soft Computing*. 1994. San Jose, CA.

19. Yang, Y., G.I. Webb, and X. Wu, *Discretization methods*, in *Data mining and knowledge discovery handbook*, O. Maimon and L. Rokach, Editors. 2010, Springer. p. 101-116.

20. Sturges, H.A., *The choice of a class interval.* Journal of the American Statistical Association, 1926. **21**(153): p. 65-66.

21. Fayyad, U.M. and K.B. Irani. *Multi-interval discretization of continuous-valued attributes for classification learning*. in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. 1993. Chambéry, France: Morgan Kaufmann 1993 ISBN 1-55860-300-X.

22. Quinlan, J.R. and R.R. L., *Inferring decision trees using the minimum description length principle.* Information and computation, 1989. **80**: p. 227-248.

23. Schildt, H., D. Coward, and Books24x7 Inc., *History and evolution of Java*, in *Java the complete reference, eighth edition*. 2011, McGraw-Hill,: New York. p. 3-16.

24. Team, R.C., *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. , 2014.

25. Weiss, M.A., *Algorithm analysis*, in *Data structures and algorithm analysis in C++*. 2006, Pearson Addison-Wesley: Boston. p. 63-89.

26. Schildt, H., D. Coward, and Books24x7 Inc., *Java util Part 1: The Collections Framework*, in *Java the complete reference, eighth edition*. 2011, McGraw-Hill,: New York. p. 453-524.

27. Shaffer, C.A. and C.A. Shaffer, *Data structures & algorithm analysis in Java*. 3rd ed. 2011, Mineola, N.Y.: Dover Publications. xix, 582 p.

28. Bache, K. and M. Lichman, *UCI Machine Learning Repository*. 2013, University of California, School of Information and Computer Science.: Irvine, CA.

29. Lu, J., et al., *MicroRNA expression profiles classify human cancers.* Nature, 2005. **435**(7043): p. 834-8.

30. Altman, E.I., *Financial ratios, discriminant analysis and the prediction of corporate bankruptcy.* The Journal of FINANCE, 1968. **23**(4): p. 589-609.