Vietnam Journal of Computer Science
Vol. 7, No. 3 (2020) 301–322
© The Author(s)
DOI: 10.1142/S2196888820500177

An Automated Unsupervised Discretization Method: A Novel Approach

Habiba Drias* and Hadjer Moulai
Computer Science Department
LRIA, USTHB, BP 32 El-Alia Bab-Ezzouar
Algiers 16111, Algeria
**hdrias@usthb.dz*

Yassine Drias
Computer Science Department
University of Algiers
Algiers 16111, Algeria

Received 1 April 2019
Accepted 1 March 2020
Published 14 April 2020

In this paper, for the first time, a novel discretization scheme is proposed aiming at enabling scalability but also at least three other strong challenges. It is based on a Left-to-Right (LR) scanning process, which partitions the input stream into intervals. This task can be implemented by an algorithm or by using a generator that builds automatically the discretization program. We focus especially on unsupervised discretization and design a method called Usupervised Left to Right Discretization (ULR-Discr). Extensive experiments were conducted using various cut-point functions on small, large and medical public datasets. First, ULR-Discr variants under different statistics are compared between themselves with the aim at observing the impact of the cut-point functions on accuracy and runtime. Then the proposed method is compared to traditional and recent techniques for classification. The result is that the classification accuracy is highly improved when using our method for discretization.

Keywords: Unsupervised discretization; massive data; classifier; data preprocessing; lexical generator.

1. Introduction

This paper follows up the paper published in Ref. 1. It deals with the unsupervised discretization method whereas previously, the supervised approach was presented.

The need to discretize continuous data is of a great utility for several applications. For instance, data discretization plays a critical role in developing classifiers,

*Corresponding author.

so as handling discrete data yields more effective outcomes than manipulating continuous data. On the other hand, it can be considered as a strategy to reduce the data size when the latter is huge, as by dividing continuous data into intervals, the cardinality of discrete values is much smaller than that of the continuous values. Another important objective is the scalability, the system treats more quickly discrete data with a smaller size than that of the continuous data. A last goal is that discrete data are easier to understand, interpret and use, in contrast to continuous data.

In general, data discretization is one of the key focuses for data preprocessing in datamining. It consists in dividing distinct values of a continuous variable into a number of disjoint ranges, each one having its own label. There are two main ways to undertake this task. The first schema, corresponding to a top-down approach, starts with the whole set of data and keeps dividing it into contiguous subsets until some termination criteria. The last subsets constitute the intervals. The second bottom-up strategy starts with a set of intervals made, respectively, by a single value and keeps merging adjacent intervals until reaching some termination conditions. The main issue is to determine a cut-point in case of division, a grouping-point in case of merging points in intervals and stopping criteria to terminate the process.

Concretely, a top-down discretization method starts with an empty list of cut-points and as the algorithm moves forward, it adds those calculated and used to partition the intervals. Prior to this process, the values are sorted, then the best cut-point is searched. Afterwards, the set of sorted values is divided into two distinct intervals according to the cut-point. The process is iterated for each range until the stop criterion is met. The division operation is handled by the means of a measure such as binning, entropy, correlation, and precision. Binning is a mechanism that consists merely in distributing the set of sorted values in several bins. These partitions can be obtained using the equal-width or equal-frequency techniques, in which a number of intervals is defined. In equal-width method, all ranges have the same width or size, whereas in equal-frequency technique, each interval contains about the same number of learning examples. Since the number of intervals is fixed at the beginning, there is no need for a stop criterion.

On the contrary, a bottom-up discretization method begins by considering each continuous value in an interval as a potential cut-point, then it merges the intervals that are similar. The corresponding algorithm is outlined in the following steps:

(1) Sort the input values.
(2) Put each value in an interval.
(3) Find the best pair of similar adjacent intervals.
(4) Merge the pair into a single interval.
(5) If the stop criterion is not satisfied then go to (3) otherwise stop.

We distinguish two kinds of discretization methods: supervised and unsupervised. Unlike a supervised method that uses the class attribute in the cut-point formulation, an unsupervised method ignores utterly the class attribute. Unsupervised methods such as equal-width and equal-frequency were the precursors of the discretization research.

In this paper, we propose a data discretization algorithm based on an original framework, called Unsupervised Left to Right Discretization (ULR-Discr). Unlike traditional methods, the approach includes both merging and division operations on intervals. As a matter of fact, the algorithmic structure is based on neither a top-down nor a bottom-up schema, but on both at the same time. It treats the data in only one pass, cutting intervals when they are dissimilar and merging them, otherwise. It thus defeats all the previous efforts in terms of time complexity. In addition, the proposed discretization schema has the ability and flexibility to modify the cut-point function. Indeed, the cut-point measure is introduced as an input to the framework, with the purpose of testing others in order to select the best one. This way, it benefits the use of a large repertoire of statistic measures.

ULR-Discr was implemented and extensive experiments were carried out on small and large public datasets as well as on medical datasets. The results are compared to those of the traditional algorithms as well as those of the recent state-of-the-art algorithms.

The rest of this paper is organized as follows. The next section provides a state of the art of the existing discretization methods. Section 3 presents an overview of Lex, a lexical generator. In Sec. 4, we describe our contribution of ULR-Discr. We explain how for the first time, Lex is used to generate a discretization algorithm and how it can explore various discretization measures. In Sec. 5, ULR-Discr is described with several discretization statistics. Section 6 reports the numerous experiments undertaken and the achieved outcomes. Comparison of the results of ULR-Discr variants under different statistics is performed. Also, ULR-Discr is tested against recent state-of-the-art algorithms. The significant results are reported and analyzed. Finally, we conclude this work and provide some future perspectives.

2. Related Work

There is a rich literature on data discretization and the efforts focused mainly on the development of methods that meet robustness and lower costs of computational. As our approach considers division and splitting operations at the same time, works on both supervised and unsupervised discretization were studied in order to invent the existing statistics used in discretization. A non-exhaustive chronological survey of the most important of them is presented as follows:

Equal Width and Equal Frequency are two unsupervised binning methods that were traditionally and are still used for their simplicity.

Chi-Merge[2] is the first effective method that proposes an approach other than division using the measurement of χ^2. It is a supervised bottom-up method that exploits the relationship between the attribute and its class to merge adjacent intervals. Initially, each distinct value of the continuous attribute is considered as an interval. Then, the χ^2 test is performed on each pair of adjacent intervals and those with the smallest value of χ^2 are merged. The process is iterated until a stop criterion is met. The value of the level of significance of χ^2 acts on the discretization and determines the number of intervals. This parameter will prevent the algorithm from creating too many intervals. The discretization stops when the χ^2 of all adjacent intervals is greater than a certain threshold, which is determined as the level of significance. The maximum number of intervals can also be used as a parameter to limit the number of constructed intervals.

In Ref. 3, based on experimentations, the authors showed that induction algorithms are more effective when data are prior discretized. They tested three methods as a preprocessing step to the C4.5 algorithm and a Naive–Bayes classifier: equal width Intervals, 1RD proposed in Ref. 4 and the entropy minimization heuristic devised in Ref. 5. The results revealed that all these methods improved the Naive–Bayes classifier accuracy especially the entropy heuristic. Increase in accuracy was also observed for C4.5 but not on all the tested datasets.

Chi2[6] is an improved version of Chi–Merge using the χ^2 measure and a second measure representing data inconsistency. The statistic χ^2 is used to discretize until an inconsistency is met in the data. Also, the number of inconsistencies is used to eliminate noise.

A comprehensive synthesis on discretization techniques is presented in Ref. 7. The authors introduced the issue of discretization as well as its history, the existing methods and its impact on classification and other applications. They concluded that important efforts have been done in the domain of discretization research but still new methods are needed.

In Ref. 8, the authors proposed a supervised discretization algorithm, called Class-Attribute Interdependence Maximization (CAIM), with the purpose to maximize the class-attribute interdependence and to minimize the number of intervals. The experiments held showed indeed the satisfaction of the two criteria when comparing the algorithm to the state-of-the-art algorithms at that time.

The authors of Ref. 9 went further in the development of the Chi2 method. They proposed an extended Chi2 algorithm with a technique that determines the predefined inconsistency rate based on the least upper bound of data misclassification error. In addition, the effect of variance in merging two intervals is considered.

The work presented in Ref. 10 proposed a top-down unsupervised method using the kernel density estimation to determine the cut-points. It uses cross-validated log-likelihood to select the maximal number of intervals. Experiments were performed to compare the proposed method to the equal-width and equal-frequency procedures. The results revealed the superiority of the suggested method.

In Ref. 11, the authors introduced an improved version of the CAIM algorithm. They considered the data distribution in the discretization formula used in CAIM in order to adjust the accuracy of the results and called their discretization algorithm Class-Attribute Contingency Coefficient (CACC). The experimental results showed that compared to CAIM, CACC yielded a better performance for classification purposes. The number of generated rules and execution time of a classifier are comparable for both algorithms.

The authors in Ref. 12 proposed an algorithm based on CACC with the aim to maintain a high interdependence between the target class and the discretized attribute for better accuracy. Experiments were performed and results were compared to those of seven methods. CACC was shown to produce the best outcomes.

In Ref. 13, the authors explored artificial intelligence tools namely, a neural network and a genetic algorithm for the determination of optimal cut-points. The cut-points are trained through a four-layer neural network and their number is optimized by a genetic algorithm. The experimental results showed a better performance compared to another method.

The author in Ref. 14 proposed an algorithm called ChiD, which is a hybrid algorithm based on Chi-merge and Chi2. He enriched the method by a criterion, the logworth to define the significance of a set of cut-points.

In Ref. 15, the authors utilized the entropy measure to discretize numeric data. A strategy based on multiple scanning of the attributes allows the selection of the attribute for the determination of the best cut-point.

The authors of Ref. 16 designed an unsupervised algorithm called ZDisc. It is based on the z-score standard deviation technique for continuous attributes on biomedical datasets. Experiments were undertaken on medical datasets and the results showed a better accuracy and also less classifier confusion for decision making process relatively to other methods.

The paper[17] presented an application of discretization in the domain of educational datamining and more precisely a prediction model for the students final grade of a particular course. They developed a discretization method called the Optimal Equal Width Binning and an over-sampling technique known as the Synthetic Minority Over-sampling TEchnique (SMOTE). They tested the application on real data and the result yields a good accuracy of the prediction model.

In Ref. 18, the authors experimented for a comparison purpose, four discretization methods, all based on entropy: a method using C4.5, the equal interval width method, the equal frequency interval method and the multiple scanning method. They used as evaluation measures, the error rate obtained by 10-fold cross-validation and the number of nodes of the tree generated by C4.5. It turned out that the multiple scanning method is the best.

Through this non-exhaustive synthesis on discretization research, we see that investigations on this topic have been regular over time and is still stimulating the interest of scientists of datamining community. We can also note that unlike supervised discretization, unsupervised discretization has been less explored. In this paper, we present an all different and novel algorithm that is capable to gather several options of the described methods in one unique platform.

3. Lex as a Tool for Numeric Discretization

Lex, a lexical analyzer[19] is a tool for automatically and rapidly implementing a lexer for a programming language or a sequence of recognizable objects. Lex is widely used in compilers construction but it is also prevalent in many areas that require patterns recognition, such as word and natural language processing. It consists in generating automatically the target program from a source of specifications containing notably, translation rules. Each translation rule includes a regular expression describing a pattern, followed by an action composed of a program fragment.

Numeric discretization consists in recognizing values of same significance or dependency and insert them in intervals. As the discretization mechanism is based on recognizing intervals of values to classify them according to their importance, Lex is an appropriate and convenient tool for handling this task. The pattern to be recognized in the discretization process is an interval and more precisely a cut-point or a grouping point. The corresponding action is executed just after the item recognition. It could be a splitting operation in case of a cut-point recognition or a merging operation in case of a grouping point recognition.

As shown in Fig. 1, we use Lex to generate automatically the discretization program from a source of specifications including rules translating data from continuous to discrete format. Afterwards, the discretization program when executed produces the intervals. Figure 2 illustrates the role of the discretization program. It takes as input, the data and the statistic threshold and computes the sequence of intervals.

Basically, the Lex source has the following format:

```
{definitions}
%%
{transition rules}
%%
{user subroutines}.
```

There are three parts delimited by the double symbol %%. The transition rules specify how to transform the source file into the target one. It may use some entities

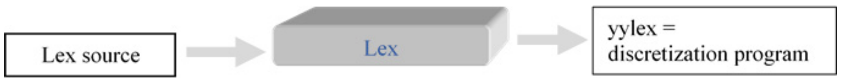

Fig. 1. Using Lex to produce the discretization program (step 1).

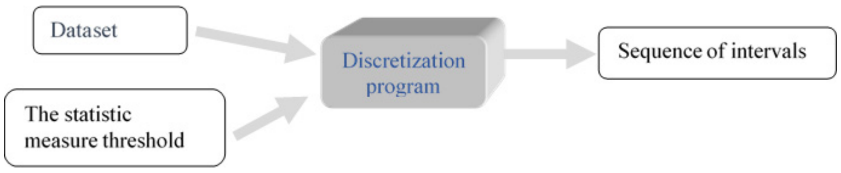

Fig. 2. Executing the discretization program for yielding the sequence of intervals (step 2).

defined in the first part of the Lex program and some procedures appearing in the last part. The first and last parts are optional as well as the second delimiter %%. In our work, transition rules have to be designed to transform the continuous values into intervals. A transition rule respects the following syntax:

Regular-expression {action}.

The regular expression builds from the objects presented in the input file, a target element. In other words, it allows the partition of the input stream data according to the target objective. The action is a program fragment to execute each time a target entity is recognized. It is written in a computer language such as C or Java. In our case, the input is represented as a sequence of pairs including a value of type *real* and its frequency of type *integer* in the dataset. Recall that the data are sorted before discretization. The Lex program we designed is outlined in Fig. 3.

Statistic is a function that calculates the measure that allows to decide to cut or to pursue the creation of an interval. For our experiments, we used χ^2, the Pearson coefficient and the Manhattan distance. For modifying the measure, we just have to change the function of the statistic that appears in the third part of the Lex source and letting the rest of the program unchanged.

```
space                [ \t\n]+
digit                [0-9]
integer              {digit}+
%%
{space}              {}
{integer}.{integer}  {value = atof(yytext) ;
                       }
{integer}            {frequency = atoi(yytext);
                      Interval2 = [(value, frequency)];
                      if (statistic(interval1, interval2) >= threshold)
                             then interval1 = interval2
                             else interval1 = merge (interval1, interval2)
                     }
%%
function statistic (interval1,interval2);
{

}

function merge (interval1, interval2);
{

}

main {
interval1 =  empty;
threshold = …;
yylex(); }
```

Fig. 3. The Lex source for generating the discretization program.

Note that this device is suitable for all kinds of discretization methods whether it be supervised or unsupervised.

4. ULR-Discr

The key idea of our approach is to scan the data in a Left-to-Right (LR) direction to create the intervals. Initially, after recognizing the two first values v_0 and v_1, the cup-point measure is calculated. If it is lower than the predefined threshold, then these two values are merged into a single interval $[v_0, v_1]$. Then on recognizing the next value, the test is performed between this new interval and the interval $[v_2]$. In case the statistic measure between v_0 and v_1 is greater than the threshold, a cut-point, which is v_1, is set. Afterwards, on recognizing the next value, the test will be performed between the intervals $[v_1]$ and $[v_2]$. The process continues for the whole input file. The maximum number of intervals, if required can be used in the experiments to tune the statistic threshold. The general framework of ULR-Discr can be summarized as follows:

(1) Sort the values and calculate their respective frequencies using the efficient well-known heapsort algorithm.
(2) Insert each value in a separate range, that is the value v_i in the interval $[v_i]$ for $i = 0$ to $n - 1$.
(3) Prepare the input file according to the following form: $v_0 f_0 v_1 f_1 \ldots v_n f_n$, where v_i is a discrete value of the attribute and f_i its frequency in the dataset.
(4) Use the Lex tool to generate the discretization program.
(5) Launch the discretization program, which will cross the input file in a LR direction, while performing the discretization.
(6) Stop the process at the end of the file.

Before launching the execution of the program, the empirical statistic threshold parameter has to be tuned in terms of either the maximum number of intervals if required, or just a level of significance.

ULR-Discr Time Complexity. Let n be the number of instances. The heapsort algorithm was selected among the sorting algorithms, to sort the numerical values because of its high performance. It was extended to count the frequencies of each distinct value and the whole operation requires a temporal complexity of $O(n \log n)$. Then, the discretization of one attribute using ULR-Discr needs traversing its values only once to create the intervals, which requires a $O(n)$ linear complexity. Therefore the total temporal complexity (including the sorting) is $O(n \log n)$.

ULR-Discr Spatial Complexity. To sort n values of an attribute, it is necessary to save them in an array of size n. After the sorting operation, the data will occupy a space of size 2^*n because each value will be followed by its frequency in the dataset. For the discretization of an attribute, one needs an array of size equal to the total number of intervals. In the worst case, this number is equal to n. The spatial complexity is therefore $O(n)$.

It is clear that the discretization time complexity of ULR-Discr is the best of all the methods found so far in the literature. It is linear because the sorted data are browsed only once to determine the intervals. The second important strength of our approach is the ability and ease allowed by the program implementation to change the cut-points measure. It is a very interesting advantage because it offers the possibility to test several convenient statistics and select the one that yields the best outcomes. Of course, the quality of the results depends also on the cut-point measure threshold. With experiments, this parameter is tuned taking into account the number of intervals.

5. Some Exploited Measures

As mentioned previously, we implemented ULR-Discr with three statistics, which are the χ^2, the coefficient of Pearson and the distance of Manhattan. Of course, other measures could be used such as the entropy and any kind of Minkowski distance.

5.1. *The χ^2 statistic*

χ^2 is a statistic measure that evaluates the relationship between two objects with regards to the classes they belong to, respectively. In discretization, it is used to measure whether two adjacent intervals are independent of the classes their values belong to or not. Its formula is given in the following equation:

$$\chi^2 = \sum_{i=1}^{i=2} \sum_{j=1}^{j=p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}, \tag{1}$$

where p is the number of classes, A_{ij} is the number of distinct values in the interval i of class j, R_i is the number of instances in the interval i, which is equal to $\sum_{j=1}^{j=p} A_{ij}$, C_j is the number of instances in class j, which is equal to $\sum_{i=1}^{i=m} A_{ij}$, n is the total number of instances, which is equal to $\sum_{j=1}^{j=p} C_j$, E_{ij} is the expected frequency of $A_{ij} = \frac{R_i * C_j}{n}$.

In unsupervised discretization, classes do not exist for objects. Formula (1) of the χ^2 is reduced to the following equation, where only the intervals are considered.

$$\chi^2 = \sum_{K} \frac{(R_K - E)^2}{E}, \tag{2}$$

where m is the number of intervals to be compared (2 in our case), R_K is the number of values in interval K, E is the expected frequency calculated as: $E = \frac{n}{\text{MaxIntervals}}$, n is the total number of values, MaxIntervals is the maximum number of intervals (fixed by the user).

5.2. *The Pearson coefficient*

The Pearson coefficient is often used to evaluate the correlation between two numeric attributes A and B. It is defined as in the following equation:

$$\tau_{A,B} = \frac{\sum_{k=1}^{k=n}(A_k - \bar{A})(B_k - \bar{B})}{(n-1)\sigma_A\sigma_B}, \tag{3}$$

where n is the number of instances, A_k is an instance of attribute A, B_k is an instance of attribute B, $\bar{A}$ and $\bar{B}$ are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B.

To calculate the Pearson coefficient between two intervals I and J, we consider the intervals in place of the attributes. One met obstacle concerns the size of the intervals, which is different from one interval to another. So, how can we import the Pearson coefficient formula from the attributes, which are of equal size? The solution we brought is to consider the interval appearing on the left of the current cut-point and the one appearing on the right with the same size. The second range is then built with the instances that appear first on the right. Once an operation of merging is performed, the algorithm will reiterate the process with the left interval containing only the last instance of the most recently built interval. The intervals' values are normalized before computing the coefficient of correlation using the following equation:

$$\tau_{I,J} = \frac{\sum_{k=1}^{k=n}(I_k - \bar{I})(J_k - \bar{J})}{(n-1)\sigma_I\sigma_J}, \tag{4}$$

where n is the number of instances, I_k is an instance of interval I, J_k is an instance of interval J, $\bar{I}$ and $\bar{J}$ are the respective means of I and J, σ_I and σ_J are the respective standard deviation of I and J.

5.3. *The distance of Manhattan*

The distance of Manhattan between two objects i and j of the same size n is the simplest form of Minkowski distance and is defined as in Formula (5), where $x_{i,l}$ and $x_{j,l}$ are, respectively, any instance of i and j.

$$d(i,j) = \sum_{l=1}^{l=n} |x_{i,l} - x_{j,l}|. \tag{5}$$

The distance between two intervals I and J is calculated considering Eq. (6), where I_k is any value or instance of interval I and J_l is any instance of interval J. $|I|$ and $|I|$ are, respectively, the cardinality of the intervals I and J.

$$d(I,J) = \sum_{k=1}^{k=|I|}\sum_{l=1}^{l=|J|} \frac{|I_k - J_l|}{|I| + |J|}. \tag{6}$$

By dividing the sum of the distances between instances of I and J by $(|I| + |J|)$, we consider the means of the distances that separate instances of these intervals. Merging two intervals I and J is undertaken only when $d(I, J)$ is lower than a fixed threshold.

With the χ^2 and the distance of Manhattan statistics, only one look-ahead instance is tested with the current interval for determining a cut-point. We can consider it as an LR(1) discretization method.

6. Experimental Results

In order to prove the effectiveness of the proposed approach, we undertook extensive experiments with the aim to compare ULR-Discr performance under different statistics. Then in a second step, the algorithm is compared to a recent state-of-the-art method. Evaluation criteria such as accuracy, speed and scalability are considered for the comparison.

6.1. *Experimental settings*

The Lex generator as well as the Java language were used for the implementation of the Lex specifications' source and the discretization program, respectively. All the experiments were performed on a machine of Processor Intel Core i5-3317U CPU @ 1.70 GHz $\times$ 4 and a RAM of 4.00 GB under the Ubuntu 14.04 LTS 64-bit operating system.

6.2. *The tested datasets*

For a comparative purpose, the same datasets, as those used in Ref. 7 although of small size, were considered in a first step. The first part of Table 1 shows these small tested datasets with their characteristics: the number of instances (#instances) and the number of attributes (#attributes). Then large datasets shown in the second part of the table were also investigated in order to test the scalability of the proposed methods. We also treated datasets belonging to the healthcare area. They appear at the end of the table.

The k-fold cross validation also called rotation estimation is used in combination with C4.5 to assess the performance of our methods. k is set to 10, in order to be able to compare with results of previous works. The principle consists in dividing the dataset into k samples, then to select one sample as a set of validation and to consider the $(k - 1)$ remaining samples as a training set. The error is computed using the C4.5 with the training set. The operation is then repeated k times by selecting each time another sample of validation among the other $(k - 1)$ samples, which were not already used as a validation for the model.

The advantage of this method is that at the end of the process, each sample is utilized exactly once as a set of validation and $(k - 1)$ times as a training set. The means of the k errors is calculated to estimate the prediction error. In addition to the

Table 1. Popular, large and medical datasets.

Dataset	#Instances	#Attributes
Australian	690	14
Breast	699	9
Heart	270	13
Vehicle	846	18
Iris	150	4
Wine	178	13
Pima	768	8
Bupa	345	6
Thyroid	215	5
Ionos	351	34
German	1000	20
Yeast	1484	8
Hypothyroid	3772	29
Abalone	4177	8
Satimage	6435	36
Handwritten digits	10992	16
Letter recognition	20000	16
Shuttle	57999	9
Appendicitis	106	7
Cleveland	297	13
Hepatitis	155	19
SPECTF heart	267	44
Breast (Wisconsin)	569	31

computed error estimation, the number of the tree nodes built by C4.5 and the time required for the discretization process are registered. For the assessment process, the following steps are undertaken:

(1) Set the cut-point threshold and the maximum number of intervals.
(2) Divide the data in two sets: the training (9/10) and the test (1/10).
(3) Apply the algorithm of discretization on the training set.
(4) Use the cut-points obtained in the previous step to discretize the test set.
(5) Call the algorithm C4.5 with the training and test sets and register the error rate as well as the number of nodes of the obtained tree.
(6) Once the 10 iterations are performed, calculate the means of the registered error rates, running time and nodes numbers.

6.3. *ULR-Discr using the χ^2 statistic*

The results of ULR-Discr using the χ^2 statistic are shown in Table 2. The six columns report respectively the name of the dataset, the empirical parameter of the threshold tuned using the maximum number of intervals, the error rate, the number of nodes computed by C4.5, the execution time in seconds and the number of intervals obtained from the discretization. We see that for a number of intervals between 9 and

Table 2. Numerical results of ULR-Discr using χ^2 statistic.

Benchmark	Threshold	Error %	#nodes	Time	#intervals
Australian	76	11.62	48.71	0.0005	10
Breast	13	4.30	30.20	0.0002	10
Heart	50	17.94	31.17	0.0005	10
Vehicle	132	29.16	240.68	0.0006	10
Iris	5	4.81	11.80	0.0002	11
Wine	105	7.43	34.53	0.0012	10
Pima	157	24.16	64.26	0.0007	10
Bupa	60	33.08	44.33	0.0004	11
Thyroid	45	4.00	18.73	0.0004	9
Ionos	71	7.44	23.68	0.0042	10
German	120	26.93	96.23	0.0004	9
Yeast	14	64.31	28.38	0.0002	15
Hypothyroid	1033	1.85	40.13	0.0006	10
Abalone	910	75.09	129.87	0.0014	9
Satimage	1300	14.30	926.08	0.0014	10
Handwritten digits	2010	6.84	1240.89	0.0007	11
Letter	1500	17.85	8242.41	0.0003	9
Shuttle	10500	0.24	123.50	0.0006	12
Appendicitis	16	11.77	19.01	0.0004	10
Cleveland	95	44.21	80.22	0.0004	10
Hepatitis	5	16.33	3.40	0.0003	10
SEPCTF heart	25	22.51	20.37	0.0009	15
Breast (Wisconsin)	140	4.80	32.03	0.0074	13

15, the threshold follows the size of the dataset; it is considerable for large datasets. The error rate seems to adhere to this rule but only for the datasets *Yeast* and *Abalone*. The execution time is very short and instantaneous, which makes the algorithm respond in real time. The number of nodes built by C4.5 indicates how the discretization process can enhance the classifier performance but this is not our concern in this work.

6.4. *ULR-Discr using the Pearson coefficient*

Table 3 shows the results for ULR-Discr using the Pearson coefficient. It shares the same major observations as those of Table 2. In addition, we notice greater values for the threshold, which can be explained by the nature of the statistic.

6.5. *ULR-Discr using the Manhattan distance*

The results of ULR-Discr using the Manhattan distance are exhibited in Table 4. As for the two previous statistics, the results seem to follow the same remarks.

As a conclusion of the three previous experiments, the results seem to be analogous. We can deduce that ULR-Discr behaves in a certain manner, independently from the statistic used for discretization. This phenomenon is especially visible for the execution time and less apparent for the accuracy rate.

Table 3. Numerical results of ULR-Discr using the Pearson coefficient.

Benchmark	Threshold	Error %	#nodes	Time	#intervals
Australian	1535	13.557	47.95	0.0005	10
Breast	1725	4.529	25.81	0.0002	9
Heart	465	19.425	20.15	0.0004	10
Vehicle	2270	28.662	270.31	0.0006	10
Iris	350	4.146	9.98	0.0002	9
Wine	820	11.376	44.07	0.0009	12
Pima	3070	27.100	94.74	0.0005	13
Bupa	690	32.804	48.97	0.0003	11
Thyroid	600	9.678	36.93	0.0003	10
Ionos	700	7.895	33.03	0.0022	10
German	500	27.969	122.99	0.0004	7
Yeast	4500	57.477	109.45	0.0003	10
Hypothyroid	150000	6.205	21.82	0.0005	10
Abalone	12200	75.029	98.33	0.0013	10
Satimage	20200	18.799	1146.84	0.0012	10
Handwritten digits	94000	9.524	1969.70	0.0007	10
Letter	9	21.551	8460.01	0.0003	10
Shuttle	999000	0.100	162.30	0.0006	10
Appendicitis	1	19.687	8.46	0.0004	10
Cleveland	500	42.598	50.82	0.0004	14
Hepatitis	350	15.950	3.97	0.0003	11
SEPCTF heart	700	20.828	1.84	0.0009	10
Breast (Wisconsin)	20000	4.833	38.64	0.0062	12

Table 4. Numerical results of ULR-Discr using the Manhattan distance.

Benchmark	Threshold	Error %	#nodes	Time	#intervals
Australian	700.0	14.23	42.61	0.050	10
Breast	0.1	4.50	25.18	0.003	5
Heart	0.1	19.46	29.70	0.003	5
Vehicle	0.1	36.81	215.47	0.012	9
Iris	4.2	6.01	4.90	0.010	10
Wine	110.0	8.66	27.16	0.052	6
Pima	0.1	35.47	5.38	0.007	10
Bupa	80.0	37.59	30.00	0.007	10
Thyroid	21.0	6.53	31.28	0.017	10
Ionos	29.0	9.38	32.59	0.755	9
German	1000.0	29.51	112.28	0.226	10
Yeast	0.1	67.53	9.50	0.143	15
Hypothyroid	150.0	2.22	37.56	0.604	15
Abalone	0.1	73.39	652.00	4.874	10
Satimage	1400.0	26.83	1286.90	139.191	10
Handwritten digits	1000.0	10.82	1857.00	196.008	10
Letter	10.0	21.45	8471.20	0.235	10
Shuttle	4000.0	0.23	197.90	162.231	10
Appendicitis	0.1	19.69	1.00	0.021	8
Cleveland	180.0	43.50	62.51	0.033	12
Hepatitis	100.0	15.92	3.25	0.008	12
SEPCTF heart	500.0	20.14	9.62	0.016	10
Breast (Wisconsin)	1.0	7.36	18.20	2.019	9

6.6. *Comparing the accuracy before and after discretization*

In this section, we show the benefit of the discretization using our method for classification purposes. We calculated the estimated error using C4.5, first without discretization and second with discretization using ULR-Discr. Table 5 exhibits the error rate before and after discretization using ULR-Disc under its best results. We observe that the accuracy is improved by our method for 16 over 23 datasets. For the other datasets, the error rate is close to each other for both cases.

Other statistics could be tested for ULR-Discr to ameliorate this result, which is right now very significant.

6.7. *Comparison between the three variants of ULR-Discr*

The results of the three versions of ULR-Discr are now compared to see the effect of the statistic measure. Table 6 reports the results of the error rate for the three variants. According to this table, ULR-Discr with the χ^2 statistic has the best error rate in 14 out of 23 datasets, followed by ULR-Discr using the Pearson coefficient with 6 out of 23 datasets. ULR-Discr using the distance of Manhattan exhibits the lowest accuracy rate with only 3 best cases over 23. The latter functions better on one large dataset and two medical datasets.

Table 5. Error rate before and after discretization under LR-Discr.

Benchmark	Before discretization	After discretization
Australian	15.28	**11.62**
Breast	4.72	**4.30**
Heart	22.16	**17.94**
Vehicle	**26.87**	28.66
Iris	4.34	**4.15**
Wine	**6.22**	7.43
Pima	26.22	**24.16**
Bupa	33.13	**32.81**
Thyroid	8.00	**4.00**
Ionos	9.14	**7.44**
German	29.50	**26.93**
Yeast	**44.54**	57.48
Hypothyroid	**0.42**	1.85
Abalone	78.84	**73.39**
Letter	**12.02**	14.30
Satimage	13.72	**6.84**
Handwritten digits	16.13	17.85
Shuttle	**0.02**	0.10
Appendicitis	14.15	**11.773**
Cleveland	51.51	**42.60**
Hepatitis	16.12	**15.92**
SPECTF heart	25.09	**20.14**
Breast (Wisconsin)	7.20	**4.80**

Table 6. Error rate for ULR-Discr using, respectively, the χ^2, the Pearson coefficient and the distance of Manhattan.

Benchmark	χ^2	Pearson	Manhattan	Best
Australian	**11.618**	13.557	14.231	**11.62**
Breast	**4.299**	4.529	4.502	**4.30**
Heart	**17.945**	19.425	19.462	**17.94**
Vehicle	29.159	**28.662**	36.814	**28.66**
Iris	4.809	**4.146**	6.008	**4.15**
Wine	**7.431**	11.376	8.665	**7.43**
Pima	**24.157**	27.1	35.468	**24.16**
Bupa	33.08	**32.808**	37.586	**32.81**
Thyroid	**4.003**	9.678	6.534	**4.00**
Ionos	**7.443**	7.895	9.377	**7.44**
German	**26.93**	27.969	29.509	**26.93**
Yeast	64.341	**57.477**	67.53	**57.48**
Hypothyroid	**1.852**	6.205	2.222	**1.85**
Abalone	75.095	75.029	**73.389**	**73.39**
Satimage	**14.304**	18.799	26.829	**14.30**
Handwritten digits	**6.845**	9.524	10.82	**6.84**
Letter	**17.847**	21.551	21.450	**17.85**
Shuttle	0.24	**0.1**	0.23	**0.10**
Appendicitis	**11.773**	19.687	19.688	**11.77**
Cleveland	44.213	**42.598**	43.503	**42.60**
Hepatitis	16.335	15.95	**15.919**	**15.92**
SEPCTF heart	22.514	20.828	**20.145**	**20.14**
Breast (Wisconsin)	**4.805**	4.833	7.364	**4.80**

It is worthy to remark that the best outcome for the error rate can be computed in real time as the response time of the three variants is instantaneous. Therefore, the best result of ULR-Discr can be achieved, independently from the considered statistic. We call this method, which is a kind of hybridization of the three methods,

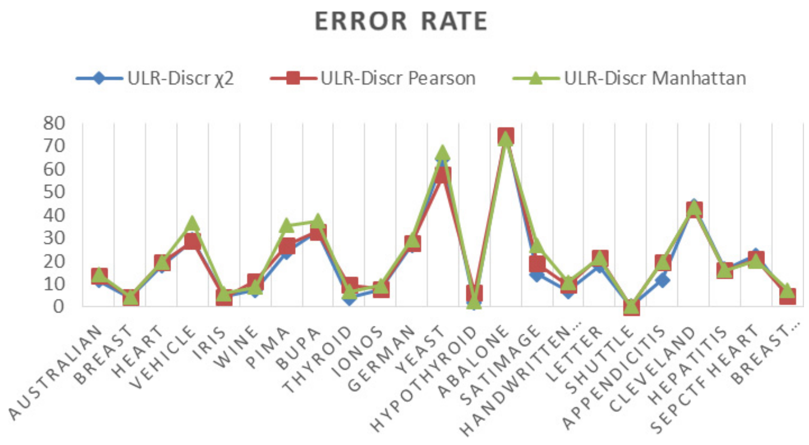

Fig. 4. Comparing error rates for ULR-Discr using χ^2, the coefficient of Pearson and the distance of Manhattan.

Table 7. Runtime for ULR-Discr using respectively the χ^2, the Pearson coefficient and the distance of Manhattan.

Benchmark	ULR-Discr χ^2	ULR-Discr Pearson	ULR-Discr Manhattan
Australian	**0.0005**	**0.0005**	0.0500
Breast	**0.0002**	**0.0002**	0.0030
Heart	0.0005	**0.0004**	0.0030
Vehicle	**0.0006**	**0.0006**	0.0120
Iris	**0.0002**	**0.0002**	0.0100
Wine	**0.0010**	**0.0009**	0.0520
Pima	0.0008	**0.0005**	0.0070
Bupa	0.0004	**0.0003**	0.0070
Thyroid	0.0004	**0.0003**	0.0170
Ionos	0.0040	**0.0020**	0.7550
German	**0.0004**	**0.0004**	0.2260
Yeast	**0.0002**	0.0003	0.1430
Hypothyroid	0.0006	**0.0005**	0.6040
Abalone	**0.0010**	**0.0010**	4.8740
Satimage	**0.0010**	**0.0010**	139.1910
Handwritten digits	**0.0007**	**0.0007**	196.0080
Letter	**0.0003**	**0.0003**	0.2350
Shuttle	**0.0006**	**0.0006**	162.2310
Appendicitis	0.0004	**0.0004**	0.0210
Cleveland	**0.0004**	**0.0004**	0.0330
Hepatitis	**0.0003**	**0.0003**	0.0080
SEPCTF heart	0.0009	**0.0009**	0.0160
Breast (Wisconsin)	**0.0004**	0.0062	2.0190

Best-ULR-Discr. The column *Best* shows the numerical values for Best-ULR-Discr. Figure 4 reinforces this finding, as it depicts almost the same graphic for the three variants of ULR-Discr.

Table 7 registers the execution time for the ULR-Discr with the three different statistics. We observe that the statistics χ^2 and Pearson coefficient yield the best execution time for ULR-Discr, although the majority of the reported times are very short and of the same order, as it is illustrated in Fig. 5. ULR-Discr with the distance of Manhattan is slower than the two others for the large datasets. This can be elucidated by the fact that the time complexity of the statistic is greater than the other two. $O(n^2)$ is needed for the Manhattan distance calculation because the distances of all pairs of instances are considered, whereas for the other two cases, only the corresponding pairs of instances are taken into account in the calculation of the statistic, which gives a $O(n)$ complexity.

6.8. *Comparison between ULR-Discr and recent state-of-the-art methods*

As evoked previously, unsupervised discretization methods are less present in the literature than the supervised ones. The most recent algorithm we found is called ZDisc[16] and was published in 2014. We used this reference for the comparison with our algorithm. Table 8 shows the error rate computed by Best-ULR-Discr and ZDisc,

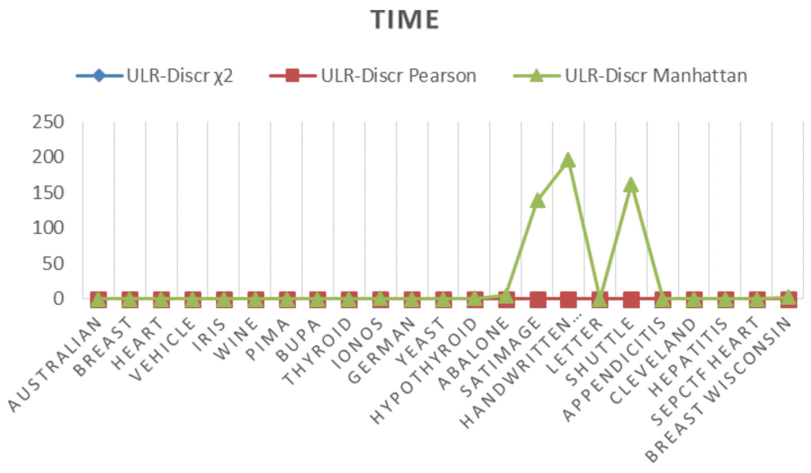

Fig. 5. Comparing runtime for the three variants of ULR-Discr.

respectively for the whole set of the considered datasets. We see clearly that the results are in favor of Best-ULR-Discr for 19 over 23 datasets. Figure 6 provides a histogram to illustrate this result, except in four cases, Best-ULR-Discr exhibits the best outcome.

Table 8. Error rate for Best-ULR-Discr and Zdisc, respectively.

Benchmark	Best-ULR-Discr	ZDisc
Australian	**11.62**	15.66
Breast	4.30	**3.72**
Heart	**17.94**	24.08
Vehicle	**28.66**	34.17
Iris	**4.15**	5.34
Wine	**7.43**	13.49
Pima	**24.16**	26.83
Bupa	**32.81**	43.19
Thyroid	**4.00**	13.59
Ionos	**7.44**	9.98
German	**26.93**	27.60
Yeast	57.48	**51.62**
Hypothyroid	**1.85**	6.31
Abalone	73.39	**46.19**
Satimage	14.30	**8.35**
Handwritten digits	**6.84**	6.87
Letter	**17.85**	39.05
Shuttle	**0.10**	5.93
Appendicitis	**11.77**	16.981
Cleveland	**42.60**	45.11
Hepatitis	**15.92**	17.53
SEPCTF heart	**20.14**	20.59
Breast (Wisconsin)	**4.80**	8.78

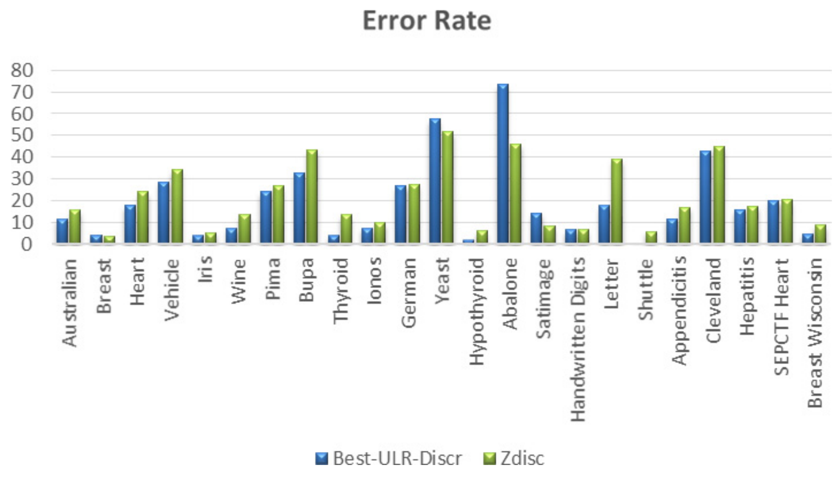

Fig. 6. Comparing error rates for Best-ULR-Discr and ZDisc.

Table 9 exhibits the results of the comparison of the runtime between ULR-Discr and ZDisc. The comparison is done more precisely with ULR-Discr using the Pearson coefficient, because the latter has the best response time. So, there is really no need to gather the best outcomes from the three versions of ULR-Discr to constitute a hybrid

Table 9. Runtime for ULR-Discr using respectively the Pearson coefficient and ZDisc.

Benchmark	ULR-Discr Pearson	ZDisc
Australian	**0.0005**	0.0157
Breast	**0.0002**	0.0159
Heart	**0.0004**	0.0114
Vehicle	**0.0006**	0.0357
Iris	**0.0002**	0.0048
Wine	**0.0009**	0.0177
Pima	**0.0005**	0.0279
Bupa	**0.0003**	0.015
Thyroid	**0.0003**	0.0038
Ionos	**0.0020**	0.0464
German	**0.0004**	0.0206
Yeast	**0.0003**	0.0220
Hypothyroid	**0.0005**	0.0626
Abalone	**0.0010**	0.0872
Satimage	**0.0010**	0.5906
Handwritten digits	**0.0007**	0.8734
Letter	**0.0003**	0.0658
Shuttle	**0.0006**	0.9488
Appendicitis	**0.0004**	0.0019
Cleveland	**0.0004**	0.0160
Hepatitis	**0.0003**	0.0050
SEPCTF heart	**0.0009**	0.0305
Breast (Wisconsin)	**0.0062**	0.0521

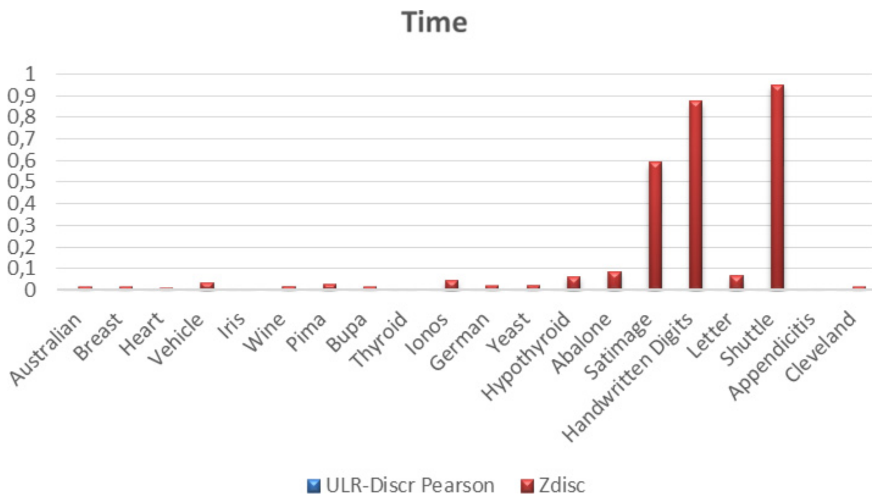

Fig. 7. Comparing time for ULR-Discr using the Pearson coefficient and ZDisc.

method. Clearly, according to this table, the results are in favor of ULR-Discr. Figure 7 expresses perfectly this result; the runtime of ULR-Discr is invisible.

At last, Table 10 summarizes the comparison for the three proposed variants of ULR-Discr with ZDisc. As an overall conclusion on these experiments, we see that, if we have to choose between the four methods from the accuracy viewpoint, ULR-Discr using the χ^2 provides the highest quality and from the side of the runtime, ULR-Discr using the Pearson coefficient as a statistic is the best. For both criteria, ULR-Discr is better than ZDisc.

Now, if we take into account Best-ULR-Discr, then Table 11 shows once more the superiority of ULR-Discr for both accuracy rate and runtime. 19 over 23 datasets were better treated by Best-ULR-Discr and for all the datasets, the runtime was widely shorter than that of ZDisc. Consequently, Best-LR-Discr outperforms ZDisc on all the datasets and for both criteria, accuracy and runtime.

Table 10. Overall comparison of the ULR-Discr variants with ZDisc.

Discretization method	#improved error %	#improved time
ULR-Discr χ^2	**14/23**	15/23
ULR-Discr Pearson coefficient	13/23	**21/23**
ULR-Discr Manhattan distance	10/23	0/23
ZDisc	7/23	0/23

Table 11. Overall comparison of Best-ULR-Discr with ZDisc.

Discretization method	#improved error %	#improved time
Best-Discr	**19/23**	23/23
ZDisc	4/23	0/23

7. Conclusions

In this paper, we proposed a discretization approach based on a novel framework that presents several strengths, among them we evoke the following:

(1) For the first time, a discretization approach based at the same time on division and merging operations is devised.
(2) This framework allows processing discretization in only one pass and hence reduces the time complexity from $O(n^2)$ to $O(n\log n)$, where n is the number of instances.
(3) The discretization algorithm can be easily implemented using a lexical generator.
(4) The approach allows the change of the discretization statistic with ease, thanks to the lexical generator. To increase the discretization accuracy, one can test several cut-point functions to retain the one that yields the best accuracy and/or best runtime.

In this study, we paid particular attention to unsupervised data discretization and developed a method called ULR-Discr. Extensive experiments were undertaken on various datasets to assess the effectiveness and efficiency of our method. Several measures have been exploited and empirical results reveal that ULR-Discr has the highest accuracy and the shortest runtime and hence, is far better than the classic algorithms and those of the recent literature. Consequently, we conclude that the proposed scheme for discretization is very promising.

For the near future, we intend to implement ULR-Discr with other statistics such as the entropy in order to enhance the accuracy rate.

References

1. H. Drias, H. Moulai and N. Rehkab, LR-SDiscr: A novel and scalable merging and splitting discretization framework using a lexical generator, *J. Information Telecommunication* **3**(2) (2019) 210–234.
2. R. Kerber, ChiMerge: Discretization of numeric attributes, in *Proc. AAAI* (Newport Beach CA, 1992), pp. 123–128.
3. J. J. Dougherty, R. Kohavi and M. Sahami, Supervised and unsupervised discretization of continuous features, in *Proc. ICML* (Tahoe City, California, USA, 1995), pp. 194–202.
4. R. C. Holte, Very simple classification rules perform well on most commonly used datasets, *Mach. Learn.* **11** (1993) 63–91.
5. U. Fayyad and K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in *Proc. Thirteenth Int. Joint Conf. Artificial Intelligence*, San Mateo CA, 1993, pp. 1022–1027.
6. H. Liu and R. Setino, Feature selection and discretization, *IEEE Trans. Knowl. Data Eng.* **9** (1997) 14.
7. H. Liu *et al.*, Discretization: An enabling technique, *Data Mining Knowl. Discov.* **6** (2002) 393–423.
8. L. Kurgan and K. J. Cios, CAIM discretization algorithm, *IEEE Trans. Knowl. Data Eng.* **16**(2) (2004) 145–153.

9. C. T. Su and J. H. Hsu, An extended chi2 algorithm for discretization of real value attributes, *IEEE Trans. Knowl. Data Eng.* **17**(3) (2005) 437–441.
10. M. Biba, F. Esposito, S. Ferilli, N. DI Mauro and T. M. A. Basile, Unsupervised discretization using kernel density estimation, in *Proc. IJCAI* (Hyderabad, India, 2007), pp. 696–701.
11. C. I. Lee, C. J. Tsai, Y. R. Yang and W. P. Yang, A top-down and greedy method for discretization of continuous attributes, in *Proc. Fourth Int. Conf. Fuzzy Systems and Knowledge Discovery*, Vol. 1 (NHaikou China, 2007), pp. 472–476.
12. C. J. Tsai, C. I. Lee and W. P. Yang, A discretization algorithm based on class-attribute contingency coefficient, *Inf. Sci.* **178** (2008) 714–731.
13. L. Shang, S. Y. Yu, X. Y. Jia and Y. S. Ji, Selection and optimization of cut-points for numeric attribute values, *Comput. Math. Appl.* **57** (2009) 1018–1023.
14. R. Bettinger, ChiD. A χ-based discretization algorithm, modern analytics, in *Proc. WUSS*, Vol. 31 (San Francisco CA, 2011).
15. J. W. Grzymala-Busse, Discretization based on entropy and multiple scanning, (2013), ISSN 1099-4300.
16. G. Madhu, T. V. Rajinikanth and A. Govardhan, Improve the classifier accuracy for continuous attributes in biomedical datasets using a new discretization method, in *Proc. Int. Conf. Information Technology and Quantitative Management*, NHaikou China, 2014, pp. 671–679.
17. S. T. Jishan, R. Raisul, I. Rashu, N. Haque and R. M. Rahman, Improving accuracy of students' final grade prediction model using optimal equal width binning and synthetic minority over-sampling technique, *Decis. Anal.* **2** (2015) 1.
18. J. W. Grzymala-Busse and T. Mroczek, A comparison of four approaches to discretization based on entropy, *Entropy* **18**(3) (2016) 69.
19. M. E. Lesk and E. Schmidt, Lex: A lexical analyzer generator, Retrieved December 12, 2016.