


## IFIT: an unsupervised discretization method based on the Ramer–Douglas–Peucker algorithm

Alev MUTLU\*, Furkan GÖZ, Orhan AKBULUT

Department of Computer Engineering, Kocaeli University, Kocaeli, Turkey

Received: 26.06.2018

Accepted/Published Online: 17.12.2019

Final Version: 15.05.2019

**Abstract:** Discretization is the process of converting continuous values into discrete values. It is a preprocessing step of several machine learning and data mining algorithms and the quality of discretization may drastically affect the performance of these algorithms. In this study we propose a discretization algorithm, namely line fitting-based discretization (IFIT), based on the Ramer–Douglas–Peucker algorithm. It is a static, univariate, unsupervised, splitting-based, global, and incremental discretization method where intervals are determined based on the Ramer–Douglas–Peucker algorithm and the quality of partitioning is assessed based on the standard error of the estimate. To evaluate the performance of the proposed method, a set of experiments are conducted on ten benchmark datasets and the achieved results are compared to those obtained by eight state-of-the-art discretization methods. Experimental results show that IFIT achieves higher predictive accuracy and produces less number of inconsistency while it generates larger number of intervals. The obtained results are also validated through Friedman’s test and Holm’s post hoc test which revealed the fact that IFIT produces discretization schemes that statistically comply both with supervised and unsupervised discretization methods.

**Key words:** Unsupervised discretization, the Ramer–Douglas–Peucker algorithm, polyline simplification, the standard error of the estimate

### 1. Introduction

In computing and statistics, discretization refers to the problem of converting continuous values into discrete values. Discretization is a crucial step of several machine learning and data mining algorithms as some algorithms work only with discrete values [1, 2], algorithms may run faster and build more accurate models with discrete values [3], and the performance of discretization may drastically affect the performance of the overall system [4]. Moreover, it was reported that discrete values are more informative and models built on such values are rather shorter and more human interpretable [5, 6]. Although there are algorithms such as support vector machines, logistic regression, and neural networks that perform better with continuous values, these methods may suffer from difficult model interpretation and long training time [7, 8].

In this work, we present a static, univariate, unsupervised, splitting-based, global, and incremental discretization method, namely IFIT, based on the Ramer–Douglas–Peucker (RDP) algorithm [9, 10] and the standard error of the estimate. The RDP algorithm is a line-simplification and point-reduction algorithm that inputs a set of continuous values representing a polyline and a tolerance value and outputs a set of values that define starting and ending values of line segments that approximate the original polyline such that the distance

\*Correspondence: alev.mutlu@kocaeli.edu.tr

between any two points is not less than the tolerance value. In this study, we assume that points returned by the RDP algorithm correspond to cut-points and values that fall between two consecutive cut-points define an interval.

Based on the tolerance value, the RDP algorithm may return different line fittings that correspond to different discretization schemes. To find the best scheme, IFIT generates multiple schemes and calculates the standard error of the estimate of each. IFIT returns the scheme whose standard error of the estimate is less than some user-defined threshold such that it is generated with the highest tolerance value. To handle different value ranges, the proposed method implements a data-normalization step where continuous values are normalized into the  $[0, 1]$  scale.

To evaluate the performance of the proposed method, ten benchmark classification datasets are discretized with two unsupervised discretization methods, namely equal-width (EW), equal-frequency (EF), six supervised discretization methods, namely 1R [11], class-attribute contingency coefficient (CACC) [12], class-attribute interdependence maximization (CAIM) [13], MODL [14], MDLP [15], Hellinger [16], and the proposed method and the number of inconsistency and the number of cut-points the methods generated are compared. To evaluate predictive accuracy, similar to [5, 12, 17–20], C4.5 and naive Bayes classifiers are trained with each discretized scheme and the predictive accuracy of the methods are compared. The experimental results show that IFIT ranks among top methods in predictive accuracy and generates schemes with low number of inconsistency and larger number of intervals. The obtained results are also validated using Friedman’s test and the Holm’s post hoc test which support these findings.

The rest of this paper is organized as follows. In Section 2, we introduce the generic discretization process, provide a classification of discretization methods, and present evaluation metrics for discretization. This section also includes a brief overview of the discretization methods we refer to while discussing the performance of the proposed method, the RDP algorithm, and the standard error of the estimate. In Section 3, we introduce the proposed method. Section 4 presents the experimental findings and the last section concludes the paper.

## 2. Background

In this section, we provide a classification of discretization methods, introduce the generic top-down discretization process, and enumerate a number of metrics to evaluate the performance of discretization methods. This section also briefly introduces the eight discretization methods we refer to while discussing the performance of the proposed method, the RDP algorithm and the standard error of the estimate.

### 2.1. Overview of discretization methods

In the literature, there exist several discretization methods and review studies [5, 18, 21, 22] that classify discretization methods based on certain aspects. Below we provide a brief taxonomy for discretization methods.

- Top-down (spitting) vs. bottom-up (merging): A discretization method is called top-down if it starts with an empty set of cut-points and populates this set by finding cut-points that best split the data. A discretization method is called bottom-up if it initially assumes each value to be discretized as a cut-point and reduces the set of cut-points by merging adjacent intervals.
- Supervised vs. unsupervised: A method is called supervised if it considers class labels of the data during the discretization process and unsupervised if all class labels are ignored.

- Dynamic vs. static: A discretization method is called dynamic if the discretization scheme is generated while a model, i.e. classification model, is being built and static if the discretization scheme is generated beforehand.
- Local vs. global: A discretization method is called local if it needs only a portion of the data to determine a cut-point. A discretization method is called global if it requires the entire dataset to determine a cut-point.
- Direct vs. incremental: A discretization method is called direct if the number of intervals is determined before the discretization process starts. A discretization method is called incremental if the number of intervals is determined during the discretization process based on some utility function.
- Univariate vs. multivariate: A discretization method is called univariate if it discretizes one feature at a time and multivariate if it discretizes all attributes simultaneously.

In Table 1, we list some recent discretization studies and their characteristics. In Table 1, the abbreviated column names stand for, in the order of appearance, top-down, bottom-up, supervised, unsupervised, dynamic, static, local, global, direct, incremental, univariate, and multivariate. A discretization method can only belong to one of the twin categories, i.e. a discretization method may either be supervised or unsupervised, but can carry the characteristics of several categories, i.e. a discretization method can be a supervised, splitting-based, and multivariate.

**Table 1.** Some recent discretization methods and their characteristics.

Method	TD	BU	Sup	Uns	Dyn	Stc	Lcl	Glb	Dir	Inc	Uv	Mv
ur-CAIM [19]		•	•			•		•		•	•	
IPD [23]		•		•		•		•	•			•
EF_Unique [24]	•			•		•	•			•	•	
LFD [25]	•			•		•		•		•		•
EMD [26]	•		•		•			•	•			•
LAIM [27]	•		•			•		•		•	•	
TD4C [28]	•		•		•		•			•		•
AEFD [20]	•			•		•		•	•		•	
SUFDA [29]	•			•		•	•			•	•	
MAX [30]		•	•		•			•		•	•	

In this study, we focus on top-down discretization. As indicated in Algorithm 1, input to a top-down discretizer is a set of continuous values and output is the set of cut-points. The first step of top-down discretization is sorting values to be discretized either in ascending or descending order. This step can be achieved in  $O(n \log n)$  time complexity if a sorting algorithm like Merge sort is employed. The next step consists of determining a cut-point that best splits the data into two intervals. Measures such as binning [11], entropy [15], dependency [19, 27], and fitness functions of genetic algorithms [26] are employed to choose cut-points. Once a cut-point is determined, values are split into two partitions with respect to the cut-point and partitions are recursively discretized until some stopping criteria are met. Stopping criteria such as reaching maximum number of intervals [11], minimum description length [15, 31], and various statistical measures [30, 32] are introduced in the literature. In Algorithm 1,  $C_f$  is a set of continuous values to be discretized and  $D_f$  is a set of cut-points

returned by the algorithm. The *sort()* method refers to any sorting algorithm, *getBestSplittingValue()* method refers to cut-point determination, and the *insert()* method inserts a cut-point into the set of cut-points. The *getLeftPart()* and *getRightPart()* methods split the data into two intervals with respect to the cut-point.

---

**Algorithm 1** Generic top-down discretization method
 

---

```

1: function DISCRETIZE( $C_f$ )
2:    $D_f = \{\}$ 
3:   sort( $C_f$ )
4:   while (Stopping criteria not satisfied) do
5:      $t = \text{getBestSplittingValue}(C_f)$ 
6:     insert( $D_f, t$ )
7:      $C_l = \text{getLeftPart}(C_f, t)$ 
8:      $C_r = \text{getRightPart}(C_f, t)$ 
9:     Discretize( $C_l$ )
10:    Discretize( $C_r$ )
11:  return  $D_f$ 

```

---

## 2.2. Performance evaluation measures for discretization

Arity, number of inconsistency, and predictive accuracy are the most commonly used metrics to evaluate performance of discretization methods. In some studies, time taken for discretization is also considered a performance metric.

- **Arity:** In the context of discretization, arity refers to the number of intervals. In the literature, it is reported that methods that return less number of intervals are better. However, these studies also emphasize a trade-off between the arity and the number inconsistency as well as a trade-off between arity and predictive accuracy [5, 33].
- **Number of inconsistency:** In the context of discretization, two instances are called inconsistent if they have the same value for every feature but belong to different classes. Assuming that a dataset contains  $k$  class labels and  $n$  instances, and  $n_1$  patterns belong to class  $c_1$ ,  $n_2$  patterns belong to class  $c_2$ , ..., and  $n_k$  patterns belong to class  $c_k$ , the number of inconsistency for this pattern is calculated by  $n - \max(n_1, n_2, \dots, n_k)$ . The total number of inconsistency is calculated by summing up number of inconsistency for each pattern. Discretization algorithms aim to become consistent by decreasing the number of inconsistency.
- **Predictive accuracy:** Discretization is carried out as a preprocessing step of machine learning and data mining algorithms that require discrete values. A discretization method is expected to generate discretization schemes that help such algorithms to build models of high quality. As an example, when the Iris dataset is discretized using equal width discretization into two bins C4.5 classifies instances with %75.33 accuracy while classification accuracy increases to %94.66 when the same dataset is discretized into three bins. As the second scheme provides higher accuracy, it is preferable over the first one. Discretization methods aim to generate schemes that increase accuracy of classification methods.

## 2.3. Overview of reference discretization methods

In this subsection, eight discretization methods which we refer to while discussing the performance of the proposed method are briefly introduced.

- **Equal-width:** It is a static, univariate, unsupervised, splitting-based, global, and direct discretization method. Equal-width method inputs arity,  $k$ , and values to be discretized,  $D$ , and discretizes the values by splitting them into  $k$  bins of almost equal size. Interval width is calculated by using Eq. (1), where  $max(D)$  and  $min(D)$  are, respectively, the maximum and the minimum values in the dataset. Interval boundaries are defined as  $[-\infty, min+w], (min+w, min+2w], \dots (min+(k-1) \times w, \infty]$ .

$$w = (max(D) - min(D))/k \quad (1)$$

- **Equal-frequency:** Similar to equal-width discretization, equal-frequency discretization is a static, univariate, unsupervised, splitting-based, global, and direct discretization method. The method inputs values to be discretized and arity. It aims to place approximately the same number of values into each bin. In equal-frequency discretization, bin size is calculated by dividing the number of values to be discretized by the number of bins.

Both for equal-width and equal-frequency discretization, the most challenging point is determining the arity. It is usually set by trying various values, or as indicated in [18] by choosing the maximum of the number of classes in the dataset and the value that corresponds to dividing the number of values in the dataset by 100.

- **1R:** It is a static, univariate, supervised, splitting-based, global, and direct discretization method introduced in [11]. In 1R, sorted values are placed into bins such that each bin contains at least six values and majority of the values in a bin have the same class label, which determines the class label of the bin. 1R does not let a bin to start with a value whose class label is the same as the class label of the previous bin but instead enforces such a value to be inserted into the previous bin. These constraints do not hold for the last bin which groups the remaining instances.
- **MDLP:** It is static, univariate, supervised, splitting-based, local, and incremental discretization method proposed in [15]. On a sorted set of continuous values,  $D$ , it selects a candidate cut-point,  $C$ , for an attribute,  $A$ , and calculates its information entropy,  $E(A, C; D)$  based on Eq. (2).

$$E(A, C; D) = \frac{|D_1|}{|D|} Ent(D_1) + \frac{|D_2|}{|D|} Ent(D_2), \quad (2)$$

where  $Ent(D_i)$  is class entropy of partition  $D_i$  defined as Eq. (3)

$$Ent(D_i) = - \sum_{j=1}^k P(C_j, D_i) \log(P(C_j, D_i)), \quad (3)$$

where  $C_j$  donates a class label and  $P(C_j, D_i)$  is the fraction of values in partition  $D_i$  that have  $C_j$  as their class label.

The method calculates information entropy of each possible cut-point and chooses the one with the lowest value.

- **CACC:** It is a static, global, incremental, supervised, and top-down discretization algorithm based on class-attribute contingency coefficient and is proposed in [12]. In CACC, sorted values are partitioned

with respect to a value whose *cacc* value is maximum. The *cacc* of a value is calculated according to Eqs. (4) and (5);

$$cacc = \sqrt{\frac{y'}{y' + M'}} \tag{4}$$

$$y' = M[(\sum_{i=1}^S \sum_{r=1}^n \frac{q_{ir}^2}{M_{i+}M_{+r}}) - 1]/\log(n), \tag{5}$$

where *M* is the total number values in data, *n* is the number of intervals, *q<sub>ir</sub>* is the number of values with class label *c<sub>i</sub>* in interval *d<sub>r</sub>*, *M<sub>i+</sub>* is the total number of values with class label *c<sub>i</sub>*, and *M<sub>+r</sub>* is the total number of values in the interval *d<sub>r</sub>*.

- **MODL:** It is a supervised, merging-based, global, and incremental discretization method that is based on the Bayesian approach and minimal description length and is introduced in [14]. It aims to maximize Eq. (6).

$$\frac{P(Model|Data)}{P(Model) \times P(Data|Model)}. \tag{6}$$

- **CAIM:** It is a supervised, top-down, global, and static discretization method described in [13]. CAIM aims to generate as few intervals as possible while maintaining class-attribute interdependence as maximum as possible. Initially, it starts with a single interval, [min(D), max(D)] and calculates class-attribute interdependency value for each value in the interval and splits the interval with respect to value with the highest class-attribute interdependency value. CAIM criterion between class label *C* and discretization variable *d* for attribute *F* is calculated as formulated in Eq. (7)

$$CAIM(C, D|F) = \frac{\sum_{r=1}^n \frac{max_r^2}{M_{+r}}}{n}, \tag{7}$$

where *n* is the number of intervals, *max<sub>r</sub>* is the maximum of the total number of continuous values belonging to the *i<sup>th</sup>* class that are within interval (*d<sub>r-1</sub>*, *d*], and *M<sub>+r</sub>* is the total number of continuous values of *F* that are in interval (*d<sub>r-1</sub>*, *d*].

- **Hellinger:** It is a supervised, bottom-up discretization method introduced in [16]. It aims to partition values into intervals such that each interval provides approximately the same amount of information content. For each cut-point, *c* adjacent to intervals *a* and *b*, it calculates entropy according to Eq. (8)

$$E(c) = E(a) - E(b) \tag{8}$$

and merges the intervals with the least information difference. The entropy of an interval is calculated according to Eq. (9)

$$E(I) = \sqrt{|\sum_i (\sqrt{p(x_i)} - \sqrt{p(x_i|I)})|} \tag{9}$$

## 2.4. The Ramer–Douglas–Peucker algorithm

The RDP algorithm is a line-simplification and point-reduction algorithm initially introduced in [9] to approximate two-dimensional curves and was extended in [10]. The algorithm has been applied to several problems such as segmenting time series [34, 35], mining spatio-temporal patterns [36, 37], and robotics [38, 39].

The algorithm inputs a set of points,  $P$ , that defines a polyline and a tolerance value,  $\epsilon$ , and returns a subset of  $P$ ,  $P'$ .  $P'$  represents the simplified polyline such that no two values of  $P'$  are less distant than  $\epsilon$  from each other. Algorithm 2 outlines the RDP algorithm. The algorithm starts with inserting the first and the last instances of the data, say  $f$  and  $l$ , into  $P'$ . Next, a value of  $P$ ,  $p$ , that is most distant from the straight line connecting  $f$  and  $l$  is retrieved. If the distance of  $p$  to the line is less than  $\epsilon$ , the algorithm terminates and returns  $P'$ , else  $p$  is inserted into  $P'$ . Recursively, the algorithm is called with  $P[f, p]$  and  $P[(p, l)]$  and  $\epsilon$  to find new  $p$  values.

---

### Algorithm 2 The Ramer–Douglas–Peucker algorithm

---

```

1: function RDP( $P$ ,  $\epsilon$ )
2:    $P'$ .insert( $P[0]$ )
3:    $P'$ .insert( $P[P.length]$ )
4:    $dmax = 0$ 
5:    $index = 0$ 
6:   for ( $i = 1$ ;  $i < P.length$ ;  $i++$ ) do
7:      $d = distance(P[i], line(P[0], P[P.length]))$ 
8:     if ( $d > dmax$ ) then
9:        $dmax = d$ 
10:       $index = i$ 
11:  if ( $dmax > \epsilon$ ) then
12:     $P'$ .insert( $P[index]$ )
13:    RDP( $P[0 \dots index]$ ,  $\epsilon$ )
14:    RDP( $P[index \dots P.length]$ ,  $\epsilon$ )
15:  return  $P'$ 

```

---

## 2.5. The standard error of the estimate

The standard error of the estimate is a measure to evaluate accuracy of predictions made with regression lines. It is calculated by dividing the sum of the squared differences (errors) between the predicted value,  $Y'$ , and the original value,  $Y$ , by number of predictions made,  $N$ . Eq. (10) formulates the standard error of the estimate.

$$\sigma_{est} = \sqrt{\frac{\sum (\vec{Y} - \vec{Y}')^2}{N}}. \quad (10)$$

## 3. IFIT: The proposed method

In this section, we firstly present the motivation behind this study and later introduce the proposed method.

### 3.1. Motivation

In unsupervised discretization, values are distributed over bins such that similar values fall into the same bin and dissimilar values are placed into different bins. In supervised discretization methods, interval boundaries are determined with respect to class labels and values with the same class label are enforced to appear in the same bin. These two observations encourage us to assume that when sorted values are plotted on 2D surface,

values should form some clusters with respect to their class labels. In Figure 1, we plot the Iris dataset. In the plot, each feature is indicated with a different marker, and class labels are indicated by different colors of a marker. As seen in Figure 1, for features *petal width* and *petal length* markers with the same color are clustered while markers that represent *sepal width* and *sepal length* are intertwined. In the literature, it is reported that *petal width* and *petal length* are more important in predicting the class of a plant [40, 41]. Hence, this benchmark dataset supports our assumption.

A further assumption we made in this study is that, when clustered with respect to class labels and sorted, similarity between ending value of a cluster and starting value of a subsequent cluster should be more than the similarity between two values that belong to the same cluster. As seen in Figure 1, there are sharp skips between clusters particularly for *petal width* and *petal length* features.

In Figure 2, line segments obtained by the RDP algorithm are superimposed over the feature values. As Figure 2 illustrates, line segments are successful in identifying class clusters particularly for *petal width* and *petal length*.

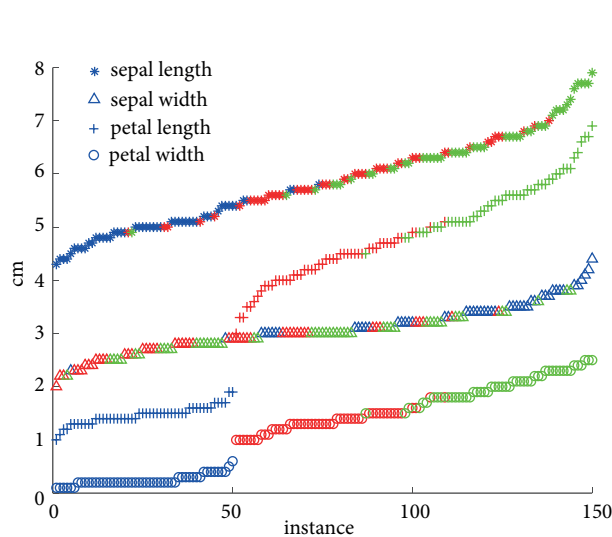


Figure 1. Features values for the Iris dataset.

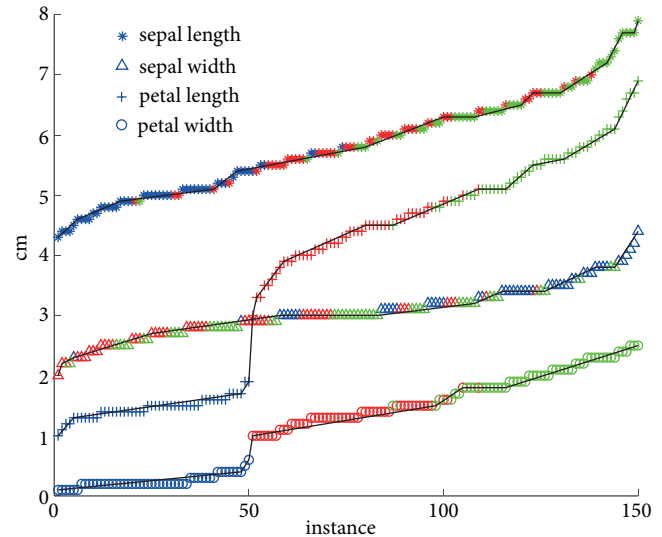


Figure 2. Segment over the features.

Based on these assumptions, when applied on a set of sorted values, we expect the RDP algorithm to return points that correspond to starting and ending values of intervals.

### 3.2. The proposed method

The proposed discretization method is static, univariate, unsupervised, splitting-based, global, and incremental. Algorithm 3 illustrates the proposed method. Input to the proposed method is a set of values,  $D$ , the maximum standard error of the estimate allowed,  $\sigma_{max}$ , and tolerance update value,  $i$ . lFIT starts with initial tolerance value  $e = 1$  and iteratively updates it by  $i$  until the standard error of the estimate for  $e$ ,  $\sigma_{est}$ , goes below  $\sigma_{max}$ . Output is a subset of  $D$  and that define the set of cut-points.

The proposed method consists of three main steps:

- **Sorting:** This step is common to several discretization methods where data is sorted in ascending order. `sortData()` method in Algorithm 3 indicates this step. Time complexity of this step is  $O(n \log n)$  if Merge sort is employed.



**Algorithm 3** The proposed method

---

```

1: function DISCRETIZERDP( $D, \sigma_{max}, i$ )
2:    $S = \text{sortData}(D)$ 
3:    $N = \text{normalize01}(S)$ 
4:    $e = 1$ 
5:   do
6:      $P = \text{RDP}(N[0, N.length], e)$ 
7:      $\sigma_{est} = \text{SEE}(N, P)$ 
8:      $e = e - i$ 
9:   while ( $\sigma_{est} > \sigma_{max}$ )
10:  return  $P$ 

```

---

- **Normalization:** In this step, sorted data is normalized into the  $[0, 1]$  range using Eq. (11), where  $d_i$  is value to be normalized,  $\max(D)$  and  $\min(D)$  are minimum and maximum values in  $D$ , respectively.

$$d'_i = \frac{d_i - \min(D)}{\max(D) - \min(D)}. \quad (11)$$

Normalization enables us to set tolerance update value and the standard error of the estimate to a fixed range as normalization scales values into notionally common sense. Complexity of this step is  $O(n)$ .

- **Discretization:** In this step data is discretized using the RDP algorithm. The discretization process is illustrated between lines 4 and 9 in Algorithm 3. In line 6, the RDP algorithm is called with various  $e$  values. For each  $e$ , the standard error of the estimate of the partitioning, referred as  $\sigma_{est}$ , is calculated. If  $\sigma_{est}$  is less than the user-set maximum allowed error of the estimate, the discretization process is terminated, otherwise  $e$  is updated and the RDP algorithm is called with a new  $e$  value.

The RDP algorithm has  $O(mn)$  complexity and  $O(n \log n)$  expected time complexity where  $n$  is the number of values in the original dataset and  $m$  is the number of values generated by the algorithm. As the number of points generated by the RDP algorithm is significantly less than the size of the input, average time complexity of this step can be assumed to be  $O(n \log n)$ .

In the implementation of the proposed method,  $e$  is iterated from 1 downward to 0. When  $e$  is 1, the Ramer–Douglas–Peucker algorithm will return the smallest and largest values of  $D$ ; hence, a discretization scheme with one interval will be generated. In this case, error of the estimate value will also be at its maximum. When  $e$  is set to 0, the RDP algorithm will return every value of  $D$  as an interval; hence, no discretization would be done and error of the estimate will be 0. In order to enforce the method to terminate with the possible minimum number of intervals, in the implementation of the method, we start with error value 1 and iteratively decrease it in order to catch the largest tolerance value that satisfies the maximum allowed error of the estimate.

#### 4. Experiments

In this section, we firstly introduce the datasets used in experiments and present the experimental settings. Next, we present the experimental results and compare the results obtained by IFIT to those obtained by discretization methods mentioned in Section 2.3. The findings are also validated via statistical analysis.

#### 4.1. Datasets and experimental settings

Table 2 lists the properties of the datasets used in the experiments. For each dataset, its name, the number of instances, and the number of attributes it has, the number of classes, and the maximum allowed standard error of the estimate are shown. The standard error of the estimate values are those that obtain the best results. As IFIT is evaluated using both C4.5 and naive Bayes classifiers, in Table 2, we report error values for both classifiers. The datasets used in the experiments are taken from (<https://sci2s.ugr.es/discretization>) and (<https://sci2s.ugr.es/keel/category.php?cat=clas>).

**Table 2.** Properties of datasets used in experiments and experimental settings.

Datasets	# Inst.	# Cnt. Att.	# Classes	e (C4.5)	e (NB)
Breast	699	9	2	0.18	0.17
Bupa	345	6	2	0.11	0.14
Heart	270	13	2	0.50	0.25
Iris	150	4	3	0.10	0.10
Ion	351	34	2	0.42	0.38
Pendigit	10992	16	10	0.20	0.11
Pima	768	8	2	0.16	0.11
Sat	6435	36	7	0.10	0.05
Vehicle	846	18	4	0.14	0.05
Wine	178	13	3	0.22	0.10

In the following subsection, we discuss and statistically analyze the performance of IFIT in terms of predictive accuracy, arity, number of inconsistency, and running time. To discretize the datasets with reference discretization methods, KEEL tool (<http://sci2s.ugr.es/keel/download.php>) is used. During our literature review, we observed that several studies report classification accuracy of their proposed methods using C4.5 and naive Bayes classifiers [5, 12, 17–20]. For this reason, in this study, predictive accuracy of IFIT is also evaluated using these two classifiers of the Weka tool (<https://www.cs.waikato.ac.nz/ml/weka/>). The experiments are conducted on a Windows computer with i7 2.8 GHz processor and 8 GB of RAM.

To statistically analyze the results, we conducted Friedman’s test and Holm’s post hoc test on the findings. Friedman’s test is two-way analysis by variance and its null hypothesis assumes that  $n$  repeated measures come from populations from the same median. If the calculated probability,  $p$ , is below the selected significance level, the null hypothesis is rejected, hence concluding that at least one method statistically differs from the others. Although Friedman’s test can reveal if methods statistically differ, it does not tell which one does. Holm’s post hoc test is used to figure out the methods that statistically differ and those that do not. In the visual representation of Holm’s post hoc test, methods are placed on an axis according to their mean ranks and the CD ruler indicates the critical difference. Methods that do not statistically differ, whose difference is less than the critical difference, are connected via a straight line. As an example, Holm’s post hoc test results plotted in Figure 3 can be interpreted as IFIT ranks the second best with CAIM in terms of C4.5 classification accuracy and does not statistically differ from the other methods. The figure also shows that 1R ranks last and statistically differs from Con. In this study, the significance level is set to 0.05.

4.2. Experimental results

In Table 3, we present the accuracy of C4.5 classifier when trained with the discretized datasets. The *Con.* column indicates results obtained by C4.5 classifier when trained with the original dataset. Values in bold indicate the highest predictive accuracy. The last two rows indicate average accuracy and average ranks of the methods.

Table 3. Predictive accuracy results for C4.5 classifier.

	IFIT	Con.	CACC	CAIM	MODL	MDLP	EF	EW	1R	Hellinger
Breast	<b>96.1</b>	94.6	96	95.6	95	94.7	94.1	94	65.5	94
Bupa	64.1	68.7	<b>69</b>	65.5	68.4	63.2	64.6	61.4	57.4	64.1
Heart	80.7	76.7	81.5	81.1	80.7	<b>81.9</b>	80.7	80.4	67.8	78.9
Iris	95.3	<b>96</b>	94	94	95.3	94	94.7	94.7	92.7	95.3
Ion	90.9	91.5	89.5	91.5	<b>92</b>	89.2	89.5	87.8	<b>92</b>	87.5
Pendigit	92.9	<b>96.6</b>	94.6	88.8	88.7	88.5	88.8	90.9	94.3	92.2
Pima	<b>76.6</b>	74.3	<b>76.6</b>	75.7	75.1	74.2	75.3	74.2	74	75.4
Sat	85.6	85.8	85.5	<b>86.1</b>	84.7	84.4	83.5	85.5	80.7	84.6
Vehicle	70.3	73.5	69	69.1	<b>73.4</b>	72.5	66.3	70	66.4	69
Wine	92.8	93.8	94.4	<b>96</b>	95.5	92.7	83.1	84.8	86	77
A. Acc.	84.53	85.15	85.01	84.34	84.88	83.53	82.06	82.37	77.68	81.8
A. Rank.	3.4	3.2	3.2	3.4	3.7	5.6	6.1	6.2	6.8	6.1

As the results show, CACC achieves the best average rank with 3.2 and IFIT shares the second best average rank with CAIM. The experimental results further show that the supervised discretization methods perform far better than the unsupervised discretization methods. Recalling that IFIT is an unsupervised discretization method, it performs much better than the supervised discretization methods, namely MODL and MDLP.

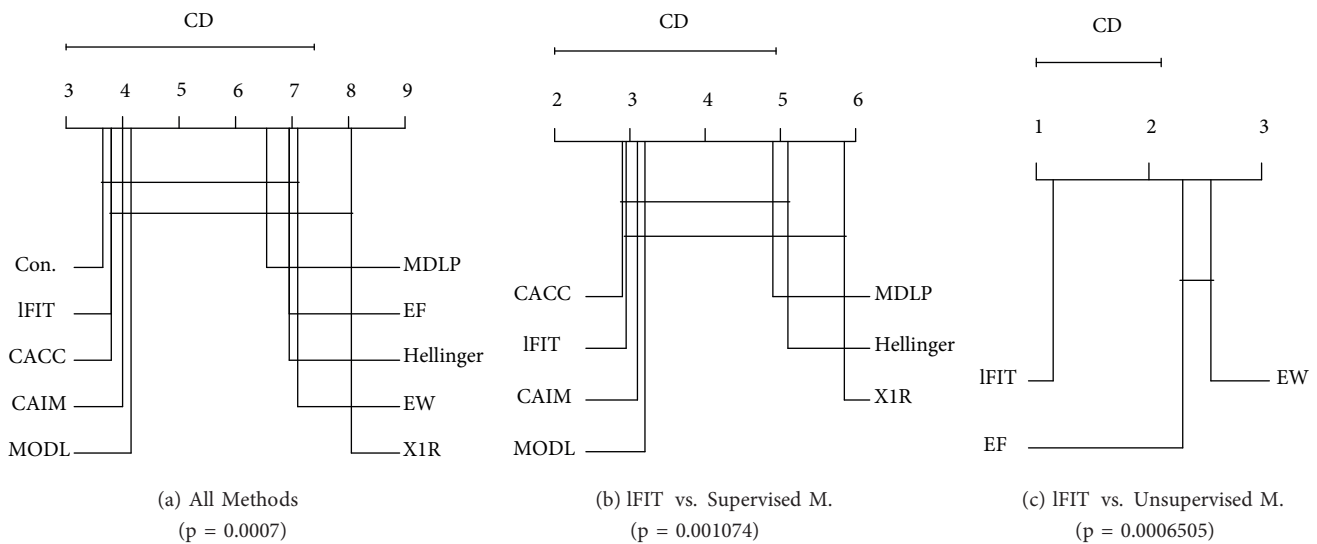


Figure 3. Holm's post hoc test results for C4.5 classification performance.

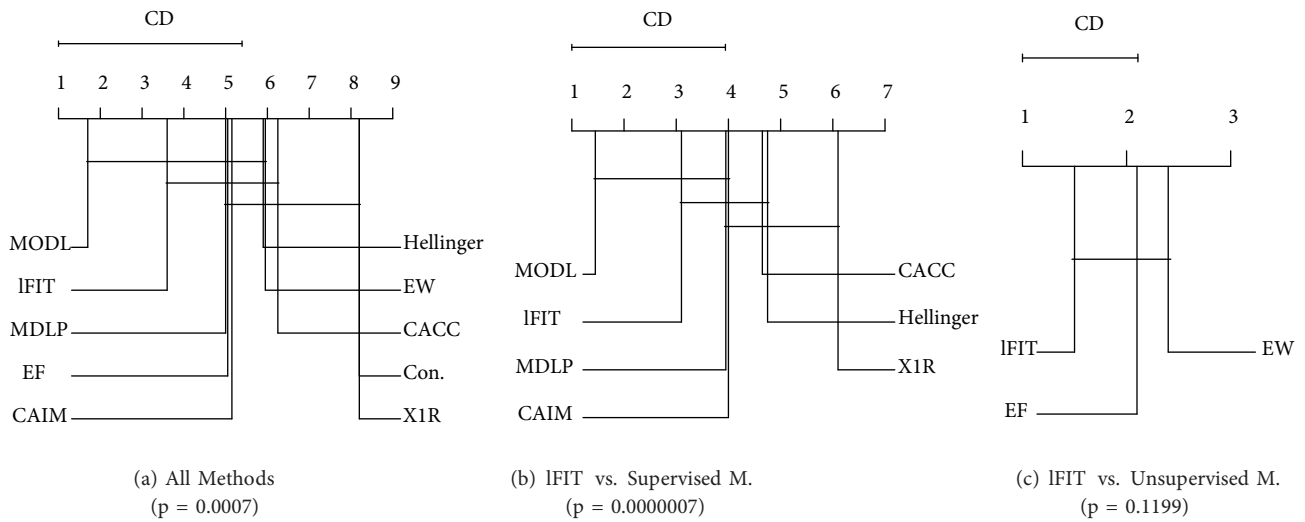
In Figure 3, we present Friedman's test and Holm's post hoc test results regarding accuracy results obtained using C4.5 classifier. When all methods are considered, Friedman's test returned a P-value of 0.0007 indicating that one method statistically differs from the others. As seen in Figure 3a, 1R statistically differs from Con., IFIT has the second best average classification accuracy with CACC, and no method statistically differs from any other. In Figure 3b, we illustrate Holm's post hoc results regarding IFIT and the supervised discretization methods. Friedman's test returned a P-value of 0.001074 for this setting and as the plot shows, CACC statistically differs from 1R. The Holm's post hoc test shows that IFIT ranks second best and no method statistically differs from IFIT. In Figure 3c, we provide statistical analysis of IFIT and the unsupervised discretization methods. For this setting, Friedman's test returned a P-value of 0.006505. As Holm's post hoc test depicts, IFIT ranks best in average accuracy and performs statistically better than EW.

In Table 4, we report classification accuracy of naive Bayes classifier when trained with the discretized datasets. Similar to Table 3, in Table 4, accuracy values in bold indicate the best classification results and the last two rows list the average accuracy and average ranks. As the results show, MODL achieves the best average rank with 1.3 while IFIT ranks second with average rank 3. The results indicate that MODL is superior over the other methods. Nonetheless, MODL is a Bayes optimal discretization method. Similar to the results of C4.5, IFIT is superior over the unsupervised discretization methods and achieves better classification results when compared to the supervised discretization methods with a naive Bayes classifier.

**Table 4.** Predictive accuracy results for naive Bayes classifier.

	IFIT	Con.	CACC	CAIM	MODL	MDLP	EF	EW	1R	Hellinger
Breast	<b>97.6</b>	96.1	96.6	97	97.3	97	97.4	97.4	65.5	97.4
Bupa	66.1	55.3	74.5	65.5	<b>77.1</b>	63.2	62.6	63.2	62.3	65.5
Heart	85.9	83.7	83.3	83.7	<b>87</b>	83.3	84.1	83.7	71.9	84.8
Iris	<b>97.3</b>	96	94.6	94.7	<b>97.3</b>	94	94	94.7	96	93.3
Ion	92	82.6	91.2	93.2	<b>97.2</b>	90.6	89.2	91.2	88.9	89.5
Pendigit	86.7	85.8	85.4	87.4	<b>88.2</b>	87.9	86.9	86.9	86.2	82.8
Pima	77.1	75.5	75.8	74	<b>83.3</b>	77.3	76	76.4	75.5	76.8
Sat	81.5	79.6	79.9	81.6	82.5	82.4	81.7	81	78.8	<b>82.8</b>
Vehicle	62.3	44.7	61.6	62.6	<b>66.3</b>	62.3	62.9	61.1	58.9	61.1
Wine	98.3	97.8	<b>98.9</b>	98.3	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	96.6	98.3	97.2
A. Acc.	84.48	79.71	84.18	83.8	87.51	83.69	83.37	83.22	78.23	83.12
A. Rank.	3	6.5	5	4	1.3	3.7	4.1	4.6	7.3	4.5

In Figure 4, we present Friedman's test and Holm's post hoc test results regarding classification accuracy of the discretization methods trained with naive Bayes classifier. Friedman's test returned a P-value of 0.0007 when all discretization methods are considered. As Figure 4a illustrates, MODL achieves the highest average classification accuracy and statistically differs from CACC and 1R while there is no other statistically differing methods. In Figure 4b, we plot Holm's post hoc test considering IFIT and the supervised discretization methods. For this setting, Friedman's test P-value was 0.0000007 indicating that at least one method statistically differs from the others. The plot indicates that MODL statistically differs from Hellinger and IFIT statistically differs from 1R. In both settings, IFIT ranks second best in average classification accuracy. When naive Bayes classification results of IFIT are compared against the unsupervised methods, Friedman's test found



**Figure 4.** Holm’s post hoc test results for naive Bayes classification performance.

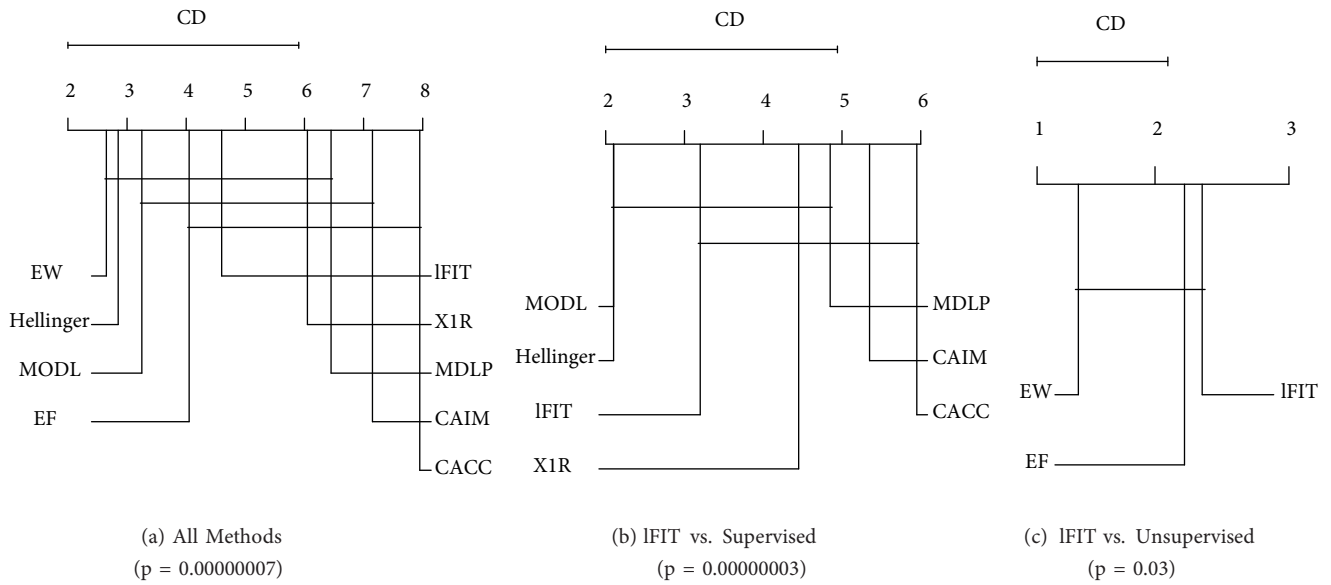
no statistically significant difference ( $P = 0.1199$ ). However, as indicated in Figure 4c, IFIT achieves the best classification accuracy when compared to the unsupervised discretization methods.

Arity is another dimension in comparing performance of discretization methods. In Table 5, we list average arity over 10-fold experiments. As the results show, IFIT has higher arity when compared to the average arity. To analyze if IFIT statistically differs from the reference discretization methods by means of arity, in Figure 5, we plot Holm’s post hoc results. As seen in Figure 5a, IFIT is connected to every other method with a straight line indicating that it does not statistically differ from them. As seen in Figure 5b, Holm’s post hoc test found no statistical difference between IFIT and the supervised discretization methods. Similarly, as indicated in Figure 5c, IFIT is not statistically different from unsupervised discretization methods by means of average arity.

**Table 5.** Arity comparison of discretization methods.

	Breast	Bupa	Heart	Iris	Ion	Pima	Vehicle	Wine	Pendigit	Sat
IFIT	2	6.08	2.5	8.5	4.11	9.50	17.02	8.23	9.50	18.9
CACC	2	5.66	2.08	2.75	3.97	2.75	3.67	2.54	3.38	3.33
CAIM	2	2	2	3	2	2	4	3	10.1	6
MODL	3.78	9.83	5	3.5	28.47	30.38	7.56	6.15	11	12.92
MDLP	3.11	1.17	1.69	3	4.26	2.13	3.83	2.85	10.31	12.44
EF	5.78	9.83	5.46	10	9.38	9.38	9.78	10	9.31	10
EW	10	10	10	10	10	10	10	10	10	10
1R	1.11	3.83	2.77	2.75	18.32	12.5	6.11	7.69	5.69	3.72
Hellinger	10	10	6.85	10	9.62	10	10	10	10	10
Average	5.15	7.37	4.93	5.94	8.91	8.89	9.48	7.39	8.81	9.70

Another criterion to evaluate performance of a discretization method is related to number of inconsistency. In Table 6, we report the number of inconsistency generated by IFIT and reference discretization methods. As



**Figure 5.** Holm's post hoc test results for arity.

seen in Table 6, except for the Bupa and Heart datasets, IFIT generates discretization schemes with less number of inconsistency compared to average number of inconsistency. The results also revealed the fact that EW and EF discretization methods schemes with least number of inconsistency. Although this may seem unexpected, this is indeed due to the fact that these methods generated schemes with the largest numbers of arity. 1R, on the other hand, generated largest number of inconsistency affecting the average number of inconsistency to a great extent. In Table 6, the last column indicates the average number of inconsistency when 1R is discarded. In this setting, the number of inconsistency generated by IFIT is well below or close to the average number on inconsistency except for the Heart dataset.

**Table 6.** Number of inconsistency comparison of discretization methods.

	Breast	Bupa	Heart	Iris	Ion	Pima	Vehicle	Wine	Pendigit	Sat	Average
IFIT	4	51	12	2	0	17	0	0	19	1	10.6
CACC	10	38	5	5	0	92	64	0	129	146	48.9
CAIM	11	97	2	5	5	153	20	0	1	4	29.8
MODL	0	16	1	4	0	0	7	0	0	0	2.8
MDLP	2	127	12	5	0	132	56	0	0	0	33.4
EF	0	0	0	0	0	0	1	0	0	0	0.1
EW	0	10	0	1	0	1	0	0	0	0	1.2
1R	241	105	37	5	0	5	13	0	179	159	74.4
Hellinger	0	28	0	1	0	17	0	0	0	2	4.8
Average	42.60	48.5	8.5	3.11	0.83	32.08	5.66	0	28.28	26.83	19.6
Avg.(-1R)	3.375	45.875	4	2.875	0.625	51.5	18.5	0	18.625	19.125	16.45

In Figure 6, we visualize Friedman's test and Holm's post hoc test results regarding inconsistency.

As Figure 6a indicates, IFIT ranks average generating larger number of inconsistency when compared to unsupervised discretization methods, MODL, and Hellinger; and less number of inconsistency when compared to 1R, MDLP, CAIM, and CACC. These findings also hold when IFIT is compared against only supervised discretization methods and unsupervised discretization methods. However, Holm’s post hoc test shows that results of IFIT do not statistically differ from reference studies.

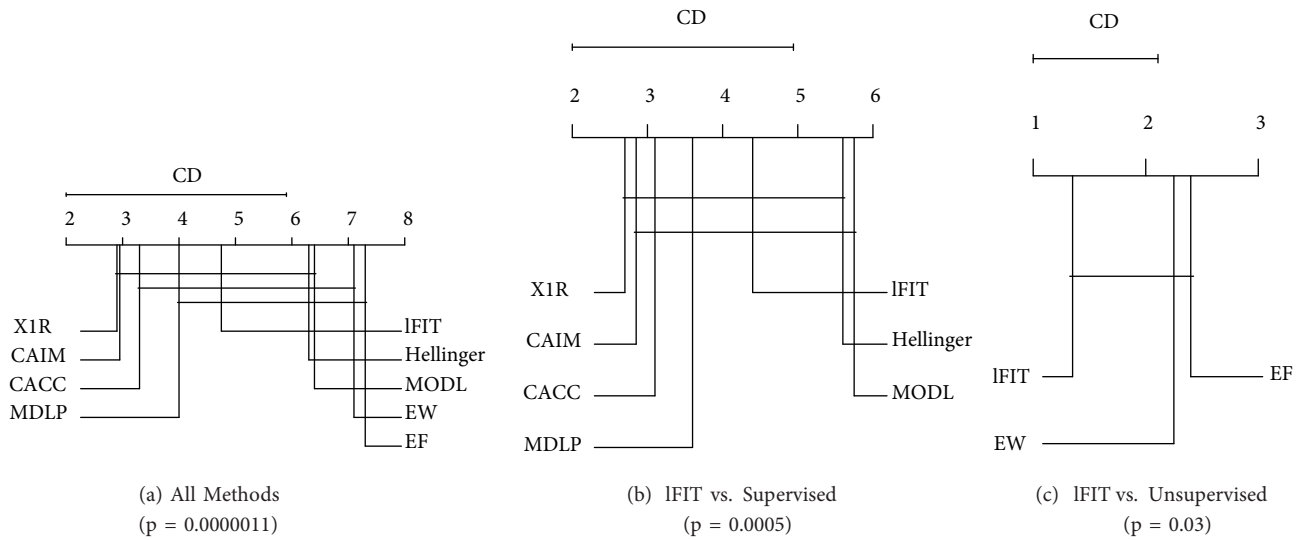


Figure 6. Holm’s post hoc test results for inconsistency.

The last dimension in evaluating performance of a discretization method is its execution time. In Table 7, we report time taken to discretize the datasets by IFIT. The obtained discretization times are comparable to those reported in [12].

Table 7. Time taken to discretize datasets (in seconds).

	Breast	Bupa	Heart	Ion	Iris	Pendigit	Sat	Pima	Vehicle	Wine
Time (s)	1.13	0.45	0.68	2.54	0.17	27.58	34.7	1.38	2.94	0.57

### 5. Conclusion

In this study, we introduced an unsupervised discretization method based on the Ramer–Douglas–Peucker algorithm and the standard error of the estimate. The Ramer–Douglas–Peucker algorithm is used to determine intervals and the standard error of the estimate is used to determine the quality of intervals. The experimental results show that IFIT generates discretization schemes that achieve promising accuracy results when compared to state-of-the-art methods. IFIT generates larger number of intervals and less number of inconsistency when compared to state-of-the-art methods.

The major limitation of the proposed method is setting the tolerance value for the Ramer–Douglas–Peucker algorithm. Future studies regarding IFIT include developing methods to automatically determine the error value based on number of intervals, number of inconsistency, and possibly some relevant statistical measures.

## References

- [1] Kaufman KA, Michalski RS. Learning from inconsistent and noisy data: the AQ18 approach. In: Ras ZW, Kowron A, editors. International Symposium on Methodologies for Intelligent Systems. Warsaw, Poland: Springer, 1999. pp. 411-419.
- [2] Cios KJ, Kurgan LA. CLIP4: Hybrid inductive machine learning algorithm that generates inequality rules. *Inf Sci* 2004; 163(1-3): 37-83.
- [3] Hsu CC, Huang YP, Chang KW. Extended naive Bayes classifier for mixed data. *Expert Syst Appl* 2008; 35(3): 1080-1083.
- [4] Tillander A. Effect of data discretization on the classification accuracy in a high-dimensional framework. *Int J Intell Syst* 2012; 27(4): 355-374.
- [5] Liu H, Hussain F, Tan CL, Dash M. Discretization: an enabling technique. *Data Min Knowl Discov* 2002; 6(4): 393-423.
- [6] Coussement K, Lessmann S, Verstraeten G. A comparative analysis of data preparation algorithms for customer churn prediction: a case study in the telecommunication industry. *Decision Support Systems* 2017; 95: 27-36.
- [7] Kotsiantis SB, Zaharakis I, Pintelas P. Supervised machine learning: a review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering* 2007; 160: 3-24.
- [8] Han J, Pei J, Kamber M. *Data Mining: Concepts and techniques*. 3rd ed. Waltham, MA, USA: Morgan Kaufmann, 2011.
- [9] Ramer U. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing* 1972; 1(3): 244-256.
- [10] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 1973; 10(2): 112-122.
- [11] Holte RC. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 1993; 11(1): 63-90.
- [12] Tsai CJ, Lee CI, Yang WP. A discretization algorithm based on class-attribute contingency coefficient. *Inf Sci* 2008; 178(3): 714-731.
- [13] Kurgan LA, Cios KJ. CAIM discretization algorithm. *IEEE Trans Knowl Data Eng* 2004; 16(2): 145-153.
- [14] Boulle M. MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning* 2006; 65(1): 131-165.
- [15] Fayyad U, Irani K. Multi-interval discretization of continuous-valued attributes for classification learning. In: Bajcsy R, editor. International Joint Conference on Artificial Intelligence. Chambéry, France: Morgan Kaufmann, 1993. pp. 1022-1029.
- [16] Lee CH. A Hellinger-based discretization method for numeric attributes in classification learning. *Knowl-Based Syst* 2007; 20(4): 419-425.
- [17] Tahan MH, Asadi S. MEMOD: a novel multivariate evolutionary multi-objective discretization. *Soft Comput* 2018; 22(1): 3013-3023.
- [18] Garcia S, Luengo J, Sáez JA, Lopez V, Herrera F. A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. *IEEE Trans Knowl Data Eng* 2013; 25(4): 734-750.
- [19] Cano A, Nguyen DT, Ventura S, Cios KJ. ur-CAIM: improved CAIM discretization for unbalanced and balanced data. *Soft Comput* 2016; 20(1): 173-188.
- [20] Jiang SY, Li X, Zheng Q, Wang LX. Approximate equal frequency discretization method. In: 2009 WRI World Congress on Computer Science and Information Engineering; March 31–April 2 2009; Los Angeles, CL, USA. IEEE Computer Society. pp. 514-518.



- [21] Kotsiantis S, Kanellopoulos D. Discretization techniques: a recent survey. *GESTS International Transactions on Computer Science and Engineering* 2006; 32(1): 47-58.
- [22] Ramirez-Gallego S, Garcia S, Mourino-Talin H, Martinez-Rego D, Bolon-Canedo V, Alonso-Betanzos A, Benitez JM, Herrera F. Data discretization: taxonomy and big data challenge. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2016; 6(1): 5-21.
- [23] Nguyen HV, Müller E, Vreeken J, Böhm K. Unsupervised interaction-preserving discretization of multivariate data. *Data Min Knowl Discov* 2014; 28(5-6): 1366-1397.
- [24] Hacibeyoglu M, Ibrahim MH. EF\_Unique: an improved version of unsupervised equal frequency discretization method. *Arabian Journal for Science and Engineering* 2018; 2018: 1-10.
- [25] Rahman MG, Islam MZ. Discretization of continuous attributes through low frequency numerical values and attribute interdependency. *Expert Syst Appl* 2016; 45: 410-423.
- [26] Ramirez-Gallego S, Garcia S, Benitez JM, Herrera F. Multivariate discretization based on evolutionary cut points selection for classification. *IEEE Trans Cybernetics* 2016; 46(3): 595-608.
- [27] Cano A, Luna JM, Gibaja EL, Ventura S. LAIM discretization for multi-label data. *Inf Sci* 2016; 330: 370-384.
- [28] Moskovitch R, Shahar Y. Classification-driven temporal discretization of multivariate time series. *Data Min Knowl Discov* 2015; 29(4): 871-913.
- [29] Abachi HM, Hosseini S, Maskouni MA, Kangavari M, Cheung NM. Statistical discretization of continuous attributes using Kolmogorov-Smirnov test. In: Wang J, Chen J, Qi J, editors. *Australasian Database Conference*. Gold Coast, QLD, Australia: Springer, 2018. pp. 309-315.
- [30] Kurtcepe M, Güvenir HA. A discretization method based on maximizing the area under receiver operating characteristic curve. *IJPRAI* 2013; 27(01):1350002.
- [31] Gupta A, Mehrotra KG, Mohan C. A clustering-based discretization for supervised learning. *Statistics & probability letters* 2010; 80(9-10): 816-824.
- [32] Sang Y, Qi H, Li K, Jin Y, Yan D, Gao S. An effective discretization method for disposing high-dimensional data. *Inf Sci* 2014; 270: 73-91.
- [33] Garcia S, Lopez V, Luengo J, Carmona CJ, Herrera FA. Preliminary study on selecting the optimal cut points in discretization by evolutionary algorithms. In: Carmona PL, Sanchez JS, editors. *International Conference on Pattern Recognition Applications and Methods*. Algarve, Portugal: SciTePress, 2012. pp. 211-216.
- [34] Keogh E, Chu S, Hart D, Pazzani M. Segmenting time series: a survey and novel approach. *Data mining in time series databases* 2004; 57: 1-21.
- [35] Fu TC. A review on time series data mining. *Eng Appl of AI* 2011; 24(1): 164-181.
- [36] Cao H, Mamoulis N, Cheung DW. Mining frequent spatio-temporal sequential patterns. In: *IEEE International Conference on Data Mining*; 27-30 November 2005; Houston, TX, USA. IEEE Computer Society. pp. 82-89.
- [37] Cao H, Wolfson O, Trajcevski G. Spatio-temporal data reduction with deterministic error bounds. *VLDB J* 2006; 15(3): 211-228.
- [38] Jiang X, Bunke H. Fast segmentation of range images into planar regions by scan line grouping. *Mach Vis Appl* 1994; 7(2): 115-122.
- [39] Gutmann JS, Fukuchi M, Fujita M. 3D perception and environment map generation for humanoid robot navigation. *I J Robotics Res* 2008; 27(10): 1117-1134.
- [40] He X, Cai D, Niyogi P. Laplacian score for feature selection. In: *Advances in Neural Information Processing Systems*. 2006. pp. 507-514.
- [41] Dash M, Choi K, Scheuermann P, Liu H. Feature selection for clustering-a filter solution. In: *IEEE International Conference on Data Mining*; 9-12 December 2002; Maebashi City, Japan. IEEE Computer Society. pp. 115-122.