The Knowledge Engineering Review, Vol. 25:4, 421–449. © Cambridge University Press, 2010
doi:10.1017/S0269888910000329

Discretization as the enabling technique for the Naïve Bayes and semi-Naïve Bayes-based classification

MARCIN J. MIZIANTY[1], LUKASZ A. KURGAN[1]
and MAREK R. OGIELA[2]

[1]*Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada;*
e-mail: mizianty@ualberta.ca, lkurgan@ece.ualberta.ca
[2]*Bio-cybernetics Laboratory, Institute of Automatics, AGH University of Science and Technology, Krakow, Poland;*
e-mail: mogiela@agh.edu.pl

Abstract

Current classification problems that concern data sets of large and increasing size require scalable classification algorithms. In this study, we concentrate on several scalable, linear complexity classifiers that include one of the top 10 voted data mining methods, Naïve Bayes (NB), and several recently proposed semi-NB classifiers. These algorithms perform front-end discretization of the continuous features since by design they work only with nominal or discrete features. We address the lack of studies that investigate the benefits and drawbacks of discretization in the context of the subsequent classification. Our comprehensive empirical study considers 12 discretizers (two unsupervised and 10 supervised), seven classifiers (two classical NB and five semi-NB), and 16 data sets. We investigate the scalability of the discretizers and show that the fastest supervised discretizers fast class-attribute interdependency maximization (FCAIM), class-attribute interdependency maximization (CAIM), and information entropy maximization (IEM) provide discretization schemes with the highest overall quality. We show that discretization improves the classification accuracy when compared against the two classical methods, NB and Flexible Naïve Bayes (FNB), executed on the raw data. The choice of the discretization algorithm impacts the significance of the improvements. The MODL, FCAIM, and CAIM methods provide statistically significant improvements, while the IEM, Class-attribute contingency coefficient (CACC), and Khiops discretizers provide moderate improvements. The most accurate classification models are generated by the Averaged one-dependence estimators (AODEsr) classifier followed by AODE and HNB (Hidden Naïve Bayes). AODEsr run on data discretized with MODL, FCAIM, and CAIM provides statistically significantly better accuracies than both the classical NB methods. The worst results are obtained with the NB, FNB, and LBR (Lazy Bayes rule) classifiers. We show that although the time to build the discretization scheme could be longer than the time to train the classifier, the completion of the entire process (to discretize data, compute the classifier, and predict test instances) is often faster than the NB-based classification of the continuous instances. This is because the time to classify test instances is an important factor that is positively influenced by discretization. The biggest positive influence, both on the accuracy and the classification time, is associated with the MODL, FCAIM, and CAIM algorithms.

1 Introduction

Application areas such as intrusion detection, market basket analysis, and genomics and proteomics, to name just a few, generate millions of data points daily. A 2003 survey reports that the biggest decision support system at that time was at 29.2 terabytes, which is about 300% larger

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

than the largest system in 2001 (Winter & Auerbach, 2004). They also report that the average size of Unix databases experienced a 6-fold, and Windows databases a 14-fold, increase compared to year 2001, and that large commercial databases average about 10 billion data points (Winter & Auerbach, 2004). This size explosion was triggered by recent advances in the low-cost storage technology and the growing interest of research and industrial communities in automated analysis of large quantities of data. Nowadays, data analysis algorithms, such as the classification algorithms, must scale with these increasing sizes.

Classification, that is, prediction of a class label of an unknown instance based on a classification model that is built from a set of training instances, is a methodology that is widely utilized in both academic and industrial settings. The instances are usually described by a set of features which can be nominal (e.g. color can be red, blue, white, etc.), ordinal (e.g. rank of a student in a class), and numerical including both interval and ratio scales (e.g. temperature or a height of a man). Both ordinal and numerical features are defined as continuous features, although ordinal features cannot be scaled, that is, we cannot judge how much better is a student ranked first when compared with a student who is ranked second or fifth. Some classification algorithms, like Naïve Bayes (NB; John & Langley, 1995), AQ (Kaufman & Michalski, 1999), CLIP (Cios & Kurgan, 2002, 2004), CN2 (Clark & Niblett, 1989), and DataSqueezer (Kurgan *et al*., 2006), work only with nominal features (and continuous features that take on a small number of values). Other classifiers that work with continuous features may perform better when dealing with nominal features (Rissanen, 1978; Catlett, 1991; Liu *et al*., 2002; Kurgan & Cios, 2004). Therefore, one of the important data preprocessing steps is discretization, which converts continuous features into discrete (nominal or ordinal) features. The key characteristic of discretization is that the number of values of a continuous feature is substantially reduced. The discretization methods were widely examined during the last two decades (Wong & Liu, 1975; Rissanen, 1978; Paterson & Niblett, 1987; Wong & Chiu, 1987; Catlett, 1991; Kerber, 1992; Fayyad & Irani, 1993; Ching *et al*., 1995; Dougherty *et al*., 1995; Liu & Setiono, 1997; Wang & Liu, 1998; Kurgan & Cios, 2001; Tay & Shen, 2002; Kurgan & Cios, 2003; Boullé, 2004; Kurgan & Cios, 2004; Liu & Wang, 2005; Mehta *et al*., 2005; Boullé, 2006; Kujala & Elomaa, 2007; Lee, 2007; Tsai *et al*., 2008). This research also addressed the impact of discretization on some of the classifiers such as decision trees including ID3 (Ching *et al*., 1995), C4.5 (Fayyad & Irani, 1992; Dougherty *et al*., 1995; Kohavi & Sahami, 1996; Liu & Setiono, 1997; Liu *et al*., 2002; Tay & Shen, 2002; Liu & Wang, 2005; Lee, 2007), and C5.0 (Tay & Shen, 2002; Kurgan & Cios, 2003, 2004; Tsai *et al*., 2008), AQ rule-based learners (Ching *et al*., 1995), CLIP4 classifiers (Cios & Kurgan, 2002; Kurgan & Cios, 2003, 2004), and nearest neighbor and logistic regression (Abraham *et al*., 2006).

As was shown in (Flores *et al*., 2007), (Lee, 2007), discretization has a positive impact on the NB classifier, that is, it helps to improve the classification performed with this classifier. This classifier is very easy to use, that is, it does not require the user to input any parameters, and provides competitive prediction accuracies (John & Langley, 1995), (Webb *et al*., 2005; Zhang *et al*., 2005). NB was included in the recently published list of the top 10 data mining algorithms due to its simplicity, elegance, and robustness (Wu *et al*., 2007). One of the most attractive features of NB is that the classification model is built in linear time with respect to the number of instances, which allows for applications to large data sets. The above advantages lead to a number of follow-up studies that proposed novel extensions to the classical NB method (the so-called semi-NB classifiers; Langley *et al*., 1992; Langley & Sage, 1994; John & Langley, 1995; Friedman *et al*., 1997; Zheng & Webb, 2000; Webb *et al*., 2005; Zhang *et al*., 2005; Abraham *et al*., 2006; Jiang & Zhang, 2006), which are also characterized by linear complexity. The abovementioned characteristics and the extent of the subsequent works related to this classifier motivated us to focus exclusively on the NB classifier and its descendants.

This classifier works only with nominal features and uses approximations (distributions) to model continuous features. As a result, usage of high-quality data discretization methods instead of the simplistic approximations would likely improve prediction performance of the NB algorithm. Dougherty *et al*. have demonstrated that coupling the classical NB with the discretization

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

method outperforms the NB that uses normal distribution to model continuous features (Dougherty *et al.*, 1995). A few more recent works also studied the impact of discretization methods on the accuracy of the classical NB models (Boullé, 2004; Flores *et al.*, 2007; Lee, 2007). Unfortunately, these studies are limited in scope and thus they do not allow one to draw strong conclusions. More specifically, they concentrate on the classical NB classifier and only a few discretization methods; they investigate the impact of discretization only on the classification accuracy, and their conclusions are based on a relatively small number of benchmarking problems. The most comprehensive work to date by Yang and Webb compares 10 discretization methods on a wide range of benchmark data sets (Yang & Webb, 2002). This study is also limited to classical NB, focuses exclusively on classification accuracy, and investigates relatively old discretizers that were developed before 2002. In contrast, we explore the impact of discretization on the classical and several modern extensions of the NB classifier; we consider a dozen popular and modern discretization methods; we investigate the impact of discretization on both the accuracy and time needed to generate the discretization scheme and the classification model and to perform the classification; and we perform the experimental tests using a large number of data sets to assure that the conclusions are generic.

One recent study explored the impact of discretization on a few NB variants (Abraham *et al.*, 2006). However, the authors investigated only three older discretization methods and their experiments were limited to a few medical data sets. In our preliminary report, we analyzed the impact of discretization on the NB classifiers but this work was performed on only seven data sets, and for eight discretizers and three NB classifiers (Mizianty *et al.*, 2008). In this study, we expanded our preliminary work by including 16 benchmark data sets, 12 discretizers, and 7 classifiers. We also performed a more detailed comparative analysis with respect to both classification accuracy and time, which allowed us to draw several interesting and sound conclusions. Our analysis is based on the Friedman *F*-test and Nemenyi test (Demšar, 2006), as well as the 5×2 cv *F*-test (Alpaydin, 1999), and compares the quality of different combinations of classifiers and discretizers over multiple data sets.

2 Background

2.1 Discretization

Discretization is a process of converting the continuous domain of a feature (including both numerical and ordered features) into a nominal domain, that is, domain with a finite number of values. This process results in the generation of a discretization scheme D for a given continuous attribute F. The scheme defines disjoint discrete subintervals bound by a pair of values (boundary points).

$$D(F) : \{[d_1; d_2], \ldots, [d_i; d_{i+1}], \ldots, [d_{n-1}; d_n]\}$$

The first step of virtually all discretization algorithms is to create a set of potential boundary points. Usually, those points are defined using unique values of the input attribute. Although the set of unique values is sometimes used as potential boundary points, the most common approach is to use all midpoints between the neighboring values and the sorted unique values set as the boundary points. In case of some discretization algorithms, for example, algorithms that are based on information entropy based criterions, the boundary points should be set between points with different class labels (Fayyad & Irani, 1993), which can be used to filter out some of the potential boundary points. This speeds up discretization since the main part of the discretization process concerns scanning of the potential boundary point set to find the best subset of boundary points. In the second step, the discretization algorithms use a statistical criterion to evaluate the quality of a given set of boundary points and to guide a search for the best-performing set of boundary points.

As proposed in Liu *et al.* (2002), discretization algorithms can be classified along five different characteristics that include *supervised* versus *unsupervised*, *static* versus *dynamic*, *global* versus *local*, *top-down (splitting)* versus *bottom-up (merging)*, and *direct* versus *incremental*.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Unsupervised algorithms do not use information about class labels and generate schemes based only on distribution of the values of the continuous attributes. *Supervised* methods use class labels and a criterion to evaluate the quality of a given scheme in the context of the known class labels.

Dynamic algorithms consider the interdependence among the features and discretize a given continuous attribute using the knowledge of other attributes (e.g. discretization performed in the C4.5 classifier (Quinlan, 1993)). Contrary to this, the *static* methods discretize each attribute separately and discretization is completed independently of the subsequent learning task. In our study, all discretization algorithms are static, whereas NB and Flexible Naïve Bayes (FNB) use the dynamic method to model continuous values.

Local methods use a subset of instances when deriving the scheme, while *global* algorithms use all instances.

When performing discretization, the *top-down*/*dividing* algorithms start with only one interval which covers all available instances and successively divide it into smaller intervals at the subsequent iterations. The *bottom-up*/*merging* algorithms start with the maximal number of subintervals (usually all midpoints are selected as the boundary points) and successively merge neighboring intervals. Since usually the final discretization scheme consists of a small number of intervals, the dividing algorithms require a smaller number of steps to converge when compared with the merging algorithms.

Direct methods require a user to decide on the number of intervals of the final scheme. In contrast, *incremental* algorithms find that number on their own. Some of the incremental methods may require the user to provide a stopping criterion to terminate the discretization process.

2.2 Discretization criterions

Each supervised discretization algorithm uses a criterion to evaluate a given discretization scheme. The most common way to store information about distribution of data points in the scheme is to use a $[k+1]x[n+1]$ dimensional matrix (called the quanta matrix), which represents an exemplary discretization scheme with n intervals, k class labels, and M training instances, see Table 1. Each following column of the matrix stores information about the number of training instances in subsequent intervals. The rows represent the number of instances with subsequent class labels. In addition, the last row and last column hold aggregated information.

Given the quanta matrix Q for the attribute F and the discretization scheme D, we can define three probability values: p_{ir}, the probability of an instance from class i having a value in

Table 1 Two-dimensional quanta matrix for attribute F and discretization scheme D where q_{ir} is the number of training instances with i-th class label which are in r-th interval, M_{i+} is the number of all training instancess with i-th class label, and M_{+r} corresponds to the number of all training instancess in r-th interval. two-dimensional quanta matrix for attribute F and discretization scheme D where q_{ir} is the number of training instances with i-th class label which are in r-th interval, M_{i+} is the number of all training instancess with i-th class label, and M_{+r} corresponds to the number of all training instancess in r-th interval

	Interval					
Class	$[d_1; d_2]$	…	$(d_r; d_{r+1}]$	…	$(d_{n-1}; d_n]$	Class total
C_1	q_{11}	…	q_{1r}	…	q_{1n}	M_{1+}
…	…	…	…	…	…	…
C_i	q_{i1}	…	q_{ir}	…	q_{in}	M_{i+}
…	…	…	…	…	…	…
C_k	q_{k1}	…	q_{kr}	…	q_{kn}	M_{k+}
Interval total	M_{+1}	…	M_{+r}	…	M_{+n}	M

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

the rth interval of F; p_{i+}, the probability of an instance belonging to ith class; and p_{+r}, the probability of an instance having a value in the rth interval of attribute F. These probabilities are defined as:

$$p_{ir} = p(C_i, D_r \mid F) = \frac{q_{ir}}{M} \tag{1}$$

$$p_{i+} = p(C_i) = \frac{M_{i+}}{M} \tag{2}$$

$$p_{+r} = p(D_r \mid F) = \frac{M_{+r}}{M} \tag{3}$$

The following sections briefly describe discretization criterions which are used by the algorithms that are considered in this study.

2.2.1 Information entropy

Information entropy is a measure of uncertainty/indetermination associated with the random variable. In the case of discretization, it represents the amount of information which is needed to describe the scheme. The lower the entropy is, the better the scheme, as we need less information to describe it. Entropy is defined as

$$\mathrm{Ent}(C, D \mid F) = \sum_{i=1}^{k} \sum_{r=1}^{n} p_{ir} \log_2 \frac{1}{p_{ir}} \tag{4}$$

where if $p_{ir} = 0$ then $p_{ir} \log_2 \frac{1}{p_{ir}} = 0$. Entropy is non-negative and reaches a maximum when all probabilities have the same values.

2.2.2 Class-attribute interdependence redundancy and class-attribute interdependence uncertainty

CAIR (class-attribute interdependence redundancy; Wong & Liu, 1975) and CAIU (class-attribute interdependence uncertainty; Huang, 1996) criterions measure interdependence between class labels and generated intervals. Both criterions are insensitive to the number of class labels and distinct values. In the case of CAIR, bigger criterion value corresponds to stronger correlation between classes and intervals, while for CAIU this relation is reversed.

CAIR criterion is given by

$$\mathrm{CAIR}(C, D \mid F) = \frac{I(C, D \mid F)}{\mathrm{Ent}(C, D \mid F)} \tag{5}$$

where $Ent(C,D \mid F)$ is the information entropy and $I(C,D \mid F)$ is the class-attribute mutual information defined as

$$I(C, D \mid F) = \sum_{i=1}^{k} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{ir}}{p_{i+} p_{+r}} \tag{6}$$

As for CAIU, the criterion is expressed by:

$$\mathrm{CAIU}(C, D \mid F) = \frac{\mathrm{INFO}(C, D \mid F)}{\mathrm{Ent}(C, D \mid F)} \tag{7}$$

where $\mathrm{Ent}(C,D \mid F)$ is the information entropy and $\mathrm{INFO}(C,D|F)$ is the class-attribute information defined as

$$\mathrm{INFO}(C, D \mid F) = \sum_{i=1}^{k} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{+r}}{p_{ir}} \tag{8}$$

2.2.3 Class-attribute interdependency maximization

CAIM (class-attribute interdependency maximization) criterion was proposed in Kurgan and Cios (2004). This criterion generates intervals that contain as many instances assigned to only one of the class labels as possible. In other words, this criterion aims to generate intervals for which a

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

dominant associated class can be found. Larger CAIM value corresponds to bigger correlation between classes and intervals. The CAIM criterion is defined as

$$\mathrm{CAIM}(C, D \mid F) = \frac{\sum_{r=1}^{n} \frac{\max_r^2}{M_{+r}}}{n} \tag{9}$$

where $\max_r$ is the maximal value of q_{ir} (for i in the range [0; S]) in the r-th interval; in other words, $\max_r$ is the number of instances of the most frequent class label in the r-th interval.

The values of this criterion range between 0 and M. Value $\max_r^2$ is divided by M_{+r} to scale this value and to reward intervals with only one dominating class. The numerator is divided by the number of intervals n to generate discretization schemes with a small number of intervals.

2.2.4 *Class-attribute contingency coefficient*

Similar to the CAIM criterion, the CACC (Class-attribute contingency coefficient) criterion aims to generate schemes that maximize interdependence between class labels and intervals (Tsai *et al.*, 2008). In contrast to CAIM, which favors schemes with a small number of intervals, CACC is more likely to accept schemes with a larger number of intervals, thus potentially further increasing interdependence.

This criterion is given by

$$\mathrm{CACC}(C, D \mid F) = \sqrt{\frac{y}{y + M}} \tag{10}$$

where y is defined as

$$y = M \frac{\sum_{i=1}^{k} \sum_{r=1}^{n} \frac{q_{ir}^2}{M_{i+} M_{+r}}}{\log(n)} \tag{11}$$

The variable y is divided by $\log(n)$ to reward schemes with a smaller number of intervals. We note that in Equation (9), division by n similarly aims at minimizing the number of intervals. The authors of the CACC criterion intentionally used $\log(n)$ instead of n to reduce the influence of this factor.

2.2.5 χ^2 *criterion*

The χ^2 criterion (Kerber, 1992) is based on the statistical χ^2test that is used to evaluate the hypothesis of class attribute independence. The χ^2 criterion value corresponds to the distance between the observed and expected frequencies of instances, and can be interpreted as the distance to the hypothesis of independence between attributes. The χ^2 criterion is defined as

$$\chi^2(C, D \mid F) = \sum_{i=1}^{k} \sum_{r=1}^{n} \frac{(q_{ir} - e_{ir})^2}{e_{ir}} \tag{12}$$

where expected frequencies e_{ir} are calculated as

$$e_{ir} = \frac{M_{i+} M_{+r}}{M} \tag{13}$$

If the hypothesis is true, then the χ^2-value is distributed like a χ^2 statistic with $(n-1) \times (k-1)$ degrees of freedom. A higher χ^2-value corresponds to lower confidence in the tested hypothesis.

2.2.6 *MODL criterion*

The MODL criterion, which has been introduced in (Boullé, 2006), combines the Bayesian approach and MDL (minimal description length) model (Rissanen, 1978). The goal of the Bayesian approach is to maximize the probability $P(\mathrm{Model} \mid \mathrm{Data})$, which, with the use of Bayesian rules and considering that $P(\mathrm{Data})$ is constant, is equivalent to maximizing

$$P(\mathrm{Model} \mid \mathrm{Data}) = P(\mathrm{Data}) * P(\mathrm{Data} \mid \mathrm{Model}) \tag{14}$$

Provided that the calculation of the probabilities $P(\mathrm{Model})$ and $P(\mathrm{Data} \mid \mathrm{Model})$ is feasible, the Bayesian approach finds an optimal model for the data.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

By applying the MDL model, the Bayesian approach is reused replacing the probabilities with their negative logarithms, which can be interpreted as Shannon code lengths. This coding problem is solved by finding a model that minimizes:

$$\text{Length}(\text{Model} \mid \text{Data}) = \text{Length}(\text{Data}) * \text{Length}(\text{Data} \mid \text{Model}) \tag{15}$$

The above problem can be calculated by finding the scheme that minimizes the following equation (Boullé, 2006):

$$\text{MODL}(C, D \mid F) = \log(M) + \log\binom{M + n - 1}{n - 1} + \sum_{r=1}^{n} \log\binom{M_{+r} + k - 1}{k - 1} + \sum_{r=1}^{n} \log\binom{M_{+r}!}{q_{i1}! q_{i2}! \ldots q_{ik}!} \tag{16}$$

Algorithms that use this criterion to search for the best schema should only compute the differences between criterion values after and before splitting an interval or merging two adjacent intervals. This is computationally more efficient than computing the criterion value from scratch.

2.3 Selected discretization algorithms

We selected a total of 12 discretization algorithms that cover various discretization approaches, see Table 2. We included several older algorithms such as two unsupervised methods, equal width and equal frequency, and four commonly used supervised methods, Maximum Entropy (ME) (Wong & Chiu, 1987), Information Entropy Maximization (IEM) (Fayyad & Irani, 1993), Paterson Niblett (Paterson & Niblett, 1987), and CADD (Ching *et al.*, 1995). The main reason for

Table 2 Comparison of the 12 discretization algorithms used in the research

Year of publication	Name (reference)	Abbreviation	Characteristics	Criterion
N/A	Equal Width	EW	Unsupervised, splitting, global, direct	N/A
N/A	Equal Frequency	EF	Unsupervised, splitting, global, direct	N/A
1987	Maximum Entropy (Wong & Chiu, 1987)	ME	Supervised, splitting, global, direct	Inf. Entropy
1987	Paterson Niblett (Paterson & Niblett, 1987)	PN	Supervised, splitting, global, direct	Inf. Entropy
1993	IEM (Fayyad & Irani, 1993)	IM	Supervised, splitting, local, incremental	Inf. Entropy
1995	CADD (Ching *et al.*, 1995)	CD	Supervised, splitting/ merging, global, direct	CAIR
2002	Modified χ^2 (Tay & Shen, 2002)	MC	Supervised, merging, global, incremental	χ^2
2003	FCAIM (Kurgan & Cios, 2003)	FC	Supervised, splitting, global, incremental	CAIM
2004	Khiops (Boullé, 2004)	Kh	Supervised, merging, global, incremental	χ^2
2004	CAIM (Kurgan & Cios, 2004)	CM	Supervised, splitting, global, incremental	CAIM
2006	MODL (Boullé, 2006)	MO	Supervised, merging, global, incremental	MODL
2008	CACC (Tsai *et al.*, 2008)	CC	Supervised, splitting, global, incremental	CACC

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

their inclusion is the fact that these methods are commonly used as a benchmark set for modern discretization algorithms (Kurgan & Cios, 2001; Liu *et al.*, 2002; Kurgan & Cios, 2003, 2004; Abraham et al., 2006; Boullé, 2006). Most of the older algorithms, except IEM, are direct, that is, they require the user to specify the final number of intervals. The newer algorithms are incremental. In the case of the direct methods, we use the following equation to determine the number of intervals:

$$d = \frac{M}{3k} \tag{17}$$

where M is the number of distinct values of F and k is the number of class labels (Wong & Chiu, 1987; Kurgan & Cios, 2004). When implementing the Paterson–Niblett algorithm, we used d and k to compute the maximum and minimum number of intervals, respectively.

The simplest algorithms are unsupervised and they include the Equal Width and Equal Frequency algorithms. Both are direct methods which need a user to determine the number of intervals in the final scheme. The Equal Width method finds a minimal and a maximal value of F and divides the corresponding interval into d equally wide intervals. The Equal Frequency algorithm sorts the values of F and computes d intervals that contain the same number of values.

ME first uses the Equal Frequency algorithm to create d intervals and then perturbs interval borders as long as the perturbation increases the entropy. At each step, only the perturbation of one border that corresponds to the largest increase in the entropy is accepted.

CADD works similarly to ME, but instead of maximizing entropy, it tries to increase the value of the CAIR criterion. It also adds one more step in which some neighboring intervals are merged.

Paterson Niblett is a splitting algorithm which starts with one big interval and successfully divides it into smaller ones. In each cycle, a cut point that corresponds to the largest entropy gain is chosen. This method is driven by the user who must determine both the minimum and maximum number of intervals. After reaching the minimal number of intervals, the algorithm checks whether the next division will improve entropy; otherwise, it terminates. If the entropy value increases, then the algorithm terminates upon reaching the maximum number of intervals.

IEM is also a top-down algorithm but it uses only the data within a given interval to decide whether the interval should be divided. After every division of an interval, the method recursively checks whether the two new subintervals should be divided. It terminates when there are no more intervals to be divided.

We also included six modern discretization algorithms including Modified χ^2 (Tay & Shen, 2002), CAIM (Kurgan & Cios, 2004), FCAIM (Kurgan & Cios, 2003), Khiops (Boullé, 2004), MODL (Boullé, 2006), and CACC (Tsai *et al.*, 2008).

Modified χ^2 is an improvement of the χ^2 algorithm (Liu & Setiono, 1997). This is a bottom-up method which in each iteration decides, based on the χ^2 test, whether the two best adjacent intervals should be merged.

The class-attribute interdependence maximization (CAIM) and Fast CAIM (FCAIM) methods are both top-down methods that iteratively add boundary points by accepting a boundary, from among all midpoints, that maximizes the value of the CAIM criterion. The difference between these algorithms is that FCAIM uses a smaller set of potential boundary points that consists of midpoints between adjacent unique values that have different class labels, whereas the CAIM algorithm uses all midpoints.

The Khiops algorithm is a bottom-up method which uses the χ^2 statistic to choose a pair of intervals that maximize the χ^2-value when merged. To shorten the execution time, this method exploits the fact that the χ^2 criterion is additive. The criterion value for each interval is stored as it will not change until this interval is merged and only the values that concern the merged intervals are updated.

The MODL algorithm uses the MODL criterion which is based on the Bayesian approach, and thus it seems to be the best suited toward application with NB classifiers. It uses a technique similar to Khiops to speed up the computations. It also applies a greedy postoptimization based on the hill-climbing algorithm to find the most optimal discretization scheme.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Finally, we include one of the most recent discretization methods, CACC, which is a top-down method based on the CAIM algorithm that uses the CACC criterion when adding boundary points.

2.4 Naïve Bayes and semi-Naïve Bayes classifiers

Classification is the task in which a training set of t instances described by n features is used to build a model that is used to predict the class label $y \in c_1,\ldots,c_k$ for a test instance $x = \langle x_1,\ldots,x_n \rangle$, where x_i is the value of the i-th feature and k is the number of class labels.

2.4.1 Naïve Bayes and Flexible Naïve Bayes

The NB classifier assumes independency of features given a class label and performs classification using:

$$\text{argmax}_y(P'(y) \prod_{i=1}^{n} P'(x_i \mid y)) \tag{18}$$

where $P(y)$ and $P(x_i \mid y)$ are estimates of the respective probabilities derived from the training set. For nominal features, the conditional probabilities correspond to the probability of the i-th feature having the value of x_i for a given class label, whereas for continuous features NB uses Gaussian distribution to model them and to estimate the probability. The maximum likelihood estimates of the mean and the standard deviation of the normal distributions are based on the sample average and standard deviation for each class label. Therefore, when the continuous features do not obey the Gaussian distribution, the estimates may lead to classification errors. To this end, a flexible NB, which improves the estimation of probability distribution for continuous features, was developed (John & Langley, 1995). In this case, the distribution is averaged over a set of Gaussian kernels, where their number equals the number of values of the i-th feature in class c_i and which have a mean equal to x_i and the same standard deviation equal to $\frac{1}{\sqrt{tc_i}}$ where tc_i corresponds to the number of instances in class c_i. While NB and FNB accept continuous features, the remaining NB-derived classifiers considered require front-end discretization of continuous features.

2.4.2 Lazy Bayes rule

Lazy Bayes rule (LBR; Zheng & Webb, 2000) is an extension to NB in which the assumption of features independence is relaxed. Instead of using all features for a given class, LBR selects a subset of features W and the independence is assumed only among this subset and the class label. In this case, the classification is performed using:

$$\text{argmax}_y(P'(y \mid W) \prod_{i=1}^{n} P'(x_i \mid y, W)) \tag{19}$$

W is selected using a heuristic wrapper that aims at minimizing error on the training data set (Zheng & Webb, 2000). The selection of the subset is delayed to the classification step and it relies on the data collected at the training step. This results in an increase in complexity, that is, the complexity of the classification step equals $O(tkn^3)$.

2.4.3 Averaged one-dependence estimators

The estimation of W in LBR is time-consuming and thus a new algorithm, averaged one-dependence estimators (AODE), which relaxes the assumption of features independence that is used in the original NB, was developed (Webb *et al.*, 2005). AODE is based on an ensemble of 1-dependence NB classifiers, that is, NB that assumes dependence on y and one feature x_i

$$\text{argmax}_y(\sum_{i:1 \le i \le n \wedge F(x_i) \ge 30} P'(y \mid x_i) \prod_{j=1}^{n} P'(x_j \mid y, x_i)) \tag{20}$$

where $F(x_i)$ is a count of the number of training instances having the attribute value x_i. To avoid false probability distributions, AODE averages only the models where $F(x_i) > 30$, a minimum sample size that is widely used in statistics.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

2.4.4 Hidden Naïve Bayes

The hidden Naïve Bayes (HNB) classifier also attempts to relax the assumption of the independence of the features (Zhang *et al.*, 2005). For each attribute, this classifier creates a hidden parent which represents the influence of all other attributes. The hidden parent node is essentially a mixture of the weighted influences from all other attributes, where the weight is used to represent the importance of a given attribute. The weight can be either computed or manually assigned by a human expert.

2.4.5 Weightily averaged one-dependence estimators

Weightily averaged one-dependence estimators (WAODE) extends the AODE algorithm by varying the impact of each feature (Jiang & Zhang, 2006). While AODE treats each tree-augmented NB equally, WAODE adds a weight for each 1-dependence NB classifier. The weight is computed using correlation between a given feature and the class variable.

2.4.6 Efficient lazy elimination for averaged one-dependence estimators

This classifier is designed upon an observation that correlated features can degrade NB's accuracy. To this end, a new technique called lazy elimination (LE), whose goal is to identify and eliminate pairs of features where one feature is a generalization of the other feature, was developed (Zheng & Webb, 2006). During the training, the LE creates a table of probability estimates and the algorithm delays the elimination to the classification step. Based on attribute values of the instance that is being classified, LE deletes all features that are related to this instance. Since the information which is computed by LE during the training step is the same as the information collected by the AODE algorithm, the LE technique was coupled with AODE. At the same time, the LE technique can be used with any Bayesian classifier.

We note that besides the abovementioned classifiers, a few other NB-based classifiers were proposed. They include selective NB (SBC; Langley & Sage, 1994), tree augmented Naive Bayes (TAN), and Super Parent TAN (SP-TAN) (Friedman *et al.*, 1997). SBC uses forward feature selection to find a suitable subset of attributes that are used to construct the NB. The results presented in (Zhang *et al.*, 2005) show that SBC is outperformed by HNB and hence the former method was not included in our study. SP-TAN is a variant of TAN in which the conditional independence between the features is relaxed and where each feature depends on the class label and one other (the so-called parent) feature. The parent feature is selected based on conditional mutual information (Keogh & Pazzani, 1999). The SP-TAN classifier was shown to provide accuracy comparable to LBR (Wang & Webb, 2002) and thus it was also excluded from our study.

The NB and semi-NB classifiers are summarized in Table 3. The table also gives the computational complexity of the training (generation of the classification model on the training data set) and classification (time to apply the classification model on a test set) performed with these classifiers. We observe that all the considered methods have linear training time with respect to the number of training instances, although some of them scale in a non-linear fashion (either quadratic or cubic) with respect to the number of features. In addition, the LBR method performs poorly in the context of the classification time.

3 Experimental results

All algorithms were implemented in JAVA. The classifiers used in the research were implemented in the WEKA workbench (Witten & Frank, 2005), while we used in-house implementations for the discretization algorithms.

We used 16 data sets from a UCI repository (Asuncion & Newman, 2007), see Table 4. We selected data sets that cover a wide range of problems including different numbers of classes (between 2 and 11), instances (between 208 and 10 992), and features (between 9 and 60). The top eight data sets include mixed feature types, that is, both continuous and nominal, while the bottom eight data sets include only continuous features.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 3 Comparison and main characteristics of the nine considered Naïve Bayes and semi-Naïve Bayes classifiers. k is the number of classes, t is the number of training instances, n is the number of attributes, and v is the average number of distinct values for an attribute; tr stands for training and cl stands for classification

Name (reference)	Abbreviation	Key characteristics	Complexity (tr/cl)
Naïve Bayes (Langley *et al.*, 1992)	NB	Minimizes the prediction error by selecting *argmax* $(y, P(y\|x))$; assumes that the features are independent given the class label; uses Gaussian distribution to model continuous features;	$O(tn)/O(kn)$
SBC (Langley & Sage, 1994)	N/A	Uses forward selection to find a good subset of attributes and then uses this subset to construct a NB;	$O(tn^3)/O(kn)$
FNB (John & Langley, 1995)	FNB	Variant of the NB algorithm which improves the estimation of probability distributions for continuous features using a set of Gaussian kernels;	$O(tn)/O(kn)$
SP-TAN (Friedman *et al.*, 1997)	N/A	Assumes that each feature depends on the class label and one other attribute (parent), which is selected based on conditional mutual information; accepts only discrete or nominal features;	$O(tkn^3)/O(kn)$
LBR (Zheng & Webb, 2000)	LB	Relaxes the assumption of feature independence by choosing a subset of features W, and minimizing the prediction error by selecting; accepts only discrete or nominal features;	$O(tn)/O(tkn^3)$
AODE (Webb *et al.*, 2005)	A	Builds a one-dependence classifier for each attribute, in which the attribute is set to be the parent of all other attributes; accepts only discrete or nominal features;	$O(tn^2)/O(kn^2)$
HNB (Zhang *et al.*, 2005)	H	creates a hidden parent for each attribute, which represents the influences from all other attributes; accepts only discrete or nominal features;	$O(tn^2 + kn^2v^2)/O(kn^2)$
WAODE (Jiang & Zhang, 2006)	WA	extends the AODE algorithm by assigning different weights to different tree augmented NB in the aggregate of AODE; accepts only discrete or nominal features;	$O(tn^2)/O(kn^2)$
AODEsr (Zheng & Webb, 2006)	Asr	extends the AODE algorithm by using LE (Lazy Elimination) technique to eliminate all related attributes at the classification time; accepts only discrete or nominal features;	$O(tn^2)/O(kn^2)$

Since WAODE and HNB classifiers cannot handle missing values, we replace them with averages (for numerical features) and modes (for nominal features). The same procedure for imputation of missing values was implemented by the authors of these classifiers (Zhang *et al.*, 2005; Jiang & Zhang, 2006).

The experiments are based on five repetitions of twofold cross-validation, as suggested in Alpaydin (1999), which results in a total of 10 folds (classifications). Each data set in each of the cross-validations was discretized by all 12 algorithms using only one (training) fold to create the discretization scheme. For each setup, that is, discretization algorithm and a data set, we computed the average CAIR (Ching *et al.*, 1995) and entropy values, as well as the average time needed to perform discretization and the average number of discretization intervals for each continuous feature. These values are used to evaluate and compare individual discretization algorithms.

Next, we use both the discretized and the raw data to evaluate the impact of discretization on the subsequently performed classification with NB and semi-NB classifiers. We use NB and FNB

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 4 Benchmark data sets used in the experiments

Data set	Abbreviation	No. continuous features	No. features	No. classes	No. instances	Average no. distinct val.
Annealing data	Anneal	6	38	6	898	23.38
Horse colic database	Colic	7	22	2	368	37.87
Credit approval	Credit	6	15	2	690	124
Cylinder bands	Cylinder	18	39	2	540	44.36
Heart disease databases cleveland	Heart	6	13	2	303	47.98
Hypothyroid disease	Hypo	7	29	4	3772	130
Thyroid disease records	Sick	7	29	2	3772	130
Vowel recognition data	Vowel	10	13	11	990	444
Glass identification	Glass	9	9	7	214	63.8
Ionosphere	Iono	34	34	2	351	121
Blocks classification	Page	10	10	5	5473	665
Pen-based recognition of handwritten digits	Pen	16	16	10	10992	99.99
Statlog (Landsat satellite)	Sat	36	36	7	6433	75.01
Image segmentation data	Seg	19	19	7	2310	451
Connectionist bench (sonar, mines vs. rocks)	Sonar	60	60	2	208	97.85
Statlog (vehicle silhouettes)	Vehicle	18	18	4	846	66.28

classifiers on the raw data and NB and the remaining considered semi-NB classifiers on the discretized data (we note that for the discrete data FNB is equivalent to NB). We train classifiers on the same fold that was used to derive the discretization scheme. For each setup, that is, classifier and a discretized/original data set, we compute average time to train the classification model (training time) on the training fold, and average accuracy and average time to use the model to classify the test samples (classification time) using the test fold. We note that training and test folds are of the same size.

The results are presented in two subsections. In the first subsection, we compare discretization methods based on the generated discretization schemes. In the second subsection, we compare and analyze the impact of discretization on the accuracies of the classifiers and the time needed to complete the entire process, that is, time to discretize the data, train a classification model, and classify the test fold.

3.1 Comparison of discretization algorithms

For each data set, the considered 12 discretization algorithms are compared based on the average CAIR and entropy values (over 10-folds using five repetitions of 2-fold cross-validation), which are used to quantify the quality of the generated discretization schemes, average running time (over 10-folds), and average number of intervals (over all continuous features and folds).

CAIR values quantify the interdependence between the discretized feature and the class labels such that higher values are associated with stronger relation between these two features. The discretization schemes with the highest CAIR values, see Table 5, are generated by CACC algorithm, followed by CAIM, FCAIM, and MODL. The worst results are obtained for Maximum Entropy, Khiops, and both unsupervised algorithms. The best entropy values, that is, higher entropy values indicate weaker relation between the discretized feature and the class labels, see Table 6, were obtained by the Paterson–Niblett and IEM algorithms. Again, the FCAIM, CAIM, and MODL algorithms scored relatively well. The worst results were obtained by the Equal Width, Maximum Entropy, and Khiops algorithms. We observe that although the results obtained with CAIR and Entropy differ, that is, these measures focus on different aspects of discretization,

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 5 The average CAIR values associated with the discretization performed by the 12 discretization algorithms on the 16 benchmark data sets; higher CAIR value indicates better discretization scheme. The values in round brackets indicate standard deviation (over the 10-folds). The values in third row show rank of a given discretization algorithm for a given data set. Values in bold indicate the best results for a given data set. Last column shows average rank of a given algorithm over all data sets. Abbreviations for data set names can be found in Table 4

Algorithm	Anneal	Colic	Credit	Cylinder	Heart	Hypo	Sick	Vowel	Glass	Iono	Page	Pen	Sat	Seg	Sonar	Vehicle	Avg. rank
Equal Width	0.058	0.023	0.029	0.009	0.037	0.053	0.023	0.092	0.127	0.076	0.042	0.125	0.201	0.168	0.036	0.055	8.25
	(0.002)	(0.004)	(0.003)	(0.001)	(0.005)	(0.005)	(0.005)	(0.002)	(0.008)	(0.004)	(0.007)	(0.001)	(0.002)	(0.001)	(0.003)	(0.003)	
	9	7	12	8	9	8	8	7	8	9	8	8	6	9	6	10	
Equal frequency	0.062	0.022	0.037	0.01	0.036	0.036	0.021	0.092	0.127	0.073	0.041	0.125	0.2	0.168	0.034	0.057	8.31
	(0.004)	(0.004)	(0.004)	(0.001)	(0.005)	(0.002)	(0.002)	(0.001)	(0.009)	(0.005)	(0.002)	(0.001)	(0.002)	(0.002)	(0.003)	(0.002)	
	8	8	9	7	10	9	9	8	7	10	9	7	8	8	7	9	
Maximum Entropy	0.012	0.01	0.034	0.006	0.022	0.026	0.014	0.086	0.073	0.068	0.035	0.12	0.159	0.159	0.023	0.044	10.75
	(0.004)	(0.003)	(0.003)	(0.001)	(0.003)	(0.001)	(0.001)	(0.001)	(0.011)	(0.004)	(0.002)	(0.001)	(0.002)	(0.002)	(0.003)	(0.002)	
	12	11	10	10	12	11	10	9	12	11	12	9	11	11	9	12	
Paterson Niblett	0.066	0.02	0.04	0.011	0.04	0.158	0.063	0.062	0.152	0.085	0.087	0.115	0.171	0.157	0.032	0.067	7.56
	(0.002)	(0.009)	(0.008)	(0.001)	(0.009)	(0.01)	(0.006)	(0.001)	(0.009)	(0.005)	(0.009)	(0.001)	(0.002)	(0.002)	(0.008)	(0.004)	
	7	9	8	6	8	2	5	12	5	7	6	10	9	12	8	7	
IEM	0.07	0.009	0.049	0.003	0.047	0.135	0.071	0.073	0.113	0.103	0.108	0.134	0.201	0.228	0.02	0.08	6.94
	(0.006)	(0.008)	(0.008)	(0.003)	(0.007)	(0.011)	(0.006)	(0.003)	(0.015)	(0.009)	(0.01)	(0.001)	(0.001)	(0.002)	(0.008)	(0.005)	
	6	12	7	12	6	6	4	11	9	4	5	2	7	5	11	4	
CADD	0.018	0.026	0.053	0.016	0.04	0.015	0.008	0.099	0.103	0.111	0.037	**0.138**	0.211	0.187	0.074	0.063	6.44
	(0.005)	(0.007)	(0.005)	(0.001)	(0.004)	(0.012)	(0.001)	(0.001)	(0.014)	(0.007)	(0.002)	**(0.002)**	(0.003)	(0.002)	(0.004)	(0.003)	
	11	6	3	2	7	12	12	5	10	3	10	**1**	4	7	2	8	
Modified χ^2	**0.088**	0.032	0.053	0.005	0.059	0.112	0.06	**0.379**	**0.272**	0.095	0.067	0.103	0.17	0.217	0.02	**0.083**	5.5
	(0.004)	(0.02)	(0.006)	(0.002)	(0.006)	(0.012)	(0.018)	**(0.001)**	**(0.009)**	(0.027)	(0.006)	(0.001)	(0.04)	(0.001)	(0.005)	**(0.005)**	
	1	5	2	11	2	7	7	**1**	**1**	5	7	12	10	6	10	**1**	
CAIM	0.076	0.035	0.052	0.015	0.053	0.138	0.074	0.104	0.179	0.087	0.139	0.131	0.212	0.229	0.039	0.074	3.5
	(0.004)	(0.009)	(0.007)	(0.001)	(0.007)	(0.009)	(0.006)	(0.002)	(0.008)	(0.004)	(0.01)	(0.001)	(0.003)	(0.002)	(0.006)	(0.004)	
	4	3	4	3	3	4	2	3	4	6	2	5	2	3	3	5	
FCAIM	0.08	0.035	0.052	0.015	0.053	0.138	0.074	0.104	0.179	0.084	0.138	0.131	0.212	0.229	0.039	0.074	3.5
	(0.004)	(0.009)	(0.007)	(0.001)	(0.007)	(0.009)	(0.006)	(0.002)	(0.008)	(0.004)	(0.01)	(0.001)	(0.003)	(0.002)	(0.006)	(0.004)	
	2	3	4	3	3	4	2	3	3	8	3	5	2	3	3	5	
Khiops	0.048	0.013	0.032	0.007	0.031	0.026	0.013	0.082	0.085	0.049	0.037	0.111	0.145	0.168	0.019	0.053	10.75
	(0.005)	(0.005)	(0.004)	(0.001)	(0.005)	(0.001)	(0.001)	(0.002)	(0.012)	(0.006)	(0.002)	(0.001)	(0.001)	(0.002)	(0.004)	(0.003)	
	10	10	11	9	11	10	11	10	11	12	11	11	12	10	12	11	
MODL	0.07	0.035	0.051	0.012	0.053	0.142	0.063	0.096	0.147	0.126	0.124	0.133	0.204	0.23	0.039	0.083	4.25
	(0.005)	(0.012)	(0.007)	(0.002)	(0.007)	(0.007)	(0.016)	(0.003)	(0.011)	(0.009)	(0.008)	(0.001)	(0.001)	(0.003)	(0.007)	(0.005)	
	5	2	6	5	5	3	6	6	6	2	4	3	5	2	5	3	
CACC	0.079	**0.056**	**0.055**	**0.027**	**0.063**	**0.16**	**0.077**	0.336	0.23	**0.127**	**0.145**	0.131	**0.214**	**0.235**	**0.079**	0.083	1.5
	(0.004)	**(0.007)**	**(0.007)**	**(0.002)**	**(0.006)**	**(0.008)**	**(0.006)**	(0.012)	(0.018)	**(0.011)**	**(0.007)**	(0.001)	**(0.002)**	**(0.003)**	**(0.007)**	(0.005)	
	3	**1**	**1**	**1**	**1**	**1**	**1**	2	2	**1**	**1**	4	**1**	**1**	**1**	2	

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 6 The average Entropy values associated with the discretization performed by the twelve discretization algorithms on the 16 benchmark datasets; lower entropy value corresponds to better discretization scheme. The values in round brackets indicate standard deviation (over the 10-folds). The values in third row show rank of a given discretization algorithm for a given data set. Values in bold indicate the best results for a given data set. Last column shows average rank of a given algorithm scored over all data sets. Abbreviations for data set names can be found in Table 4

Algorithm	Anneal	Colic	Credit	Cylinder	Heart	Hypo	Sick	Vowel	Glass	Iono	Page	Pen	Sat	Seg	Sonar	Vehicle	Avg. rank
Equal width	2.202	2.788	3.178	2.674	3.169	2.153	2.776	6.299	3.353	4.175	2.561	5.549	4.056	4.715	4.314	3.745	7.81
	(0.064)	(0.042)	(0.059)	(0.056)	(0.074)	(0.171)	(0.214)	(0.015)	(0.091)	(0.05)	(0.069)	(0.007)	(0.014)	(0.012)	(0.055)	(0.032)	
	6	9	8	8	10	9	8	7	5	10	7	8	6	7	9	8	
Equal frequency	2.417	3.275	4.406	3.124	3.491	3.256	4.093	6.659	4.069	4.674	4.981	5.565	4.082	5.728	4.872	4.014	10.38
	(0.067)	(0.054)	(0.029)	(0.028)	(0.038)	(0.043)	(0.028)	(0.008)	(0.034)	(0.068)	(0.021)	(0.008)	(0.011)	(0.018)	(0.025)	(0.021)	
	10	12	11	11	11	10	10	9	11	12	11	9	7	10	11	11	
Maximum Entropy	1.849	3.024	4.581	3.245	3.504	3.843	4.716	6.694	3.504	4.654	5.631	4.722	3.826	5.875	4.893	3.964	9.44
	(0.125)	(0.058)	(0.036)	(0.023)	(0.035)	(0.037)	(0.028)	(0.01)	(0.086)	(0.082)	(0.026)	(0.008)	(0.008)	(0.016)	(0.022)	(0.033)	
	2	11	12	12	12	11	12	10	8	11	12	2	3	11	12	10	
Paterson Niblett	2.183	1.268	**1.287**	1.16	**1.325**	**0.576**	**0.407**	4.425	2.73	**1.264**	**0.71**	4.841	**3.25**	**3.288**	1.253	3.293	2
	(0.145)	(0.065)	**(0.046)**	(0.045)	**(0.055)**	**(0.029)**	**(0.012)**	(0.007)	(0.103)	**(0.044)**	**(0.033)**	(0.028)	**(0.059)**	**(0.006)**	(0.03)	(0.052)	
	5	2	**1**	3	**1**	**1**	**1**	2	3	**1**	**1**	3	**1**	**1**	3	3	
IEM	1.919	**1.032**	1.532	**1**	1.437	0.909	0.847	**4.196**	**2.418**	1.584	1.743	5.383	4.492	4.219	**1.145**	**2.957**	2.88
	(0.122)	**(0.076)**	(0.141)	**(0.036)**	(0.082)	(0.09)	(0.071)	**(0.064)**	**(0.096)**	(0.119)	(0.096)	(0.032)	(0.035)	(0.038)	**(0.059)**	**(0.063)**	
	4	**1**	2	**1**	2	2	2	**1**	**1**	4	6	4	10	4	**1**	**1**	
CADD	**1.727**	2.754	4.166	2.827	3.112	1.835	3.663	6.609	3.344	4.028	4.472	**4.618**	3.539	5.53	4.532	3.741	7.06
	(0.142)	(0.072)	(0.065)	(0.037)	(0.08)	(0.211)	(0.076)	(0.009)	(0.072)	(0.057)	(0.109)	**(0.024)**	(0.031)	(0.021)	(0.045)	(0.049)	
	1	8	10	9	9	8	9	8	4	9	9	**1**	2	9	10	7	
Modified χ^2	2.875	1.492	2.527	1.057	1.872	1.495	2.032	8.63	5.217	1.747	4.389	8.138	5.789	7.814	1.179	3.851	7.94
	(0.075)	(0.425)	(0.419)	(0.057)	(0.251)	(0.275)	(0.61)	(0.011)	(0.053)	(0.395)	(0.429)	(0.006)	(1.559)	(0.02)	(0.041)	(0.124)	
	12	3	7	2	6	7	7	12	12	5	8	12	11	12	2	9	
CAIM	2.273	1.533	1.79	1.367	1.77	1.206	0.888	5.588	3.481	1.396	1.24	5.518	4.048	4.123	1.719	3.44	4.25
	(0.079)	(0.085)	(0.026)	(0.105)	(0.098)	(0.113)	(0.062)	(0.101)	(0.102)	(0.105)	(0.057)	(0.027)	(0.019)	(0.017)	(0.078)	(0.061)	
	8	4	3	4	4	4	3	4	7	3	3	6	4	2	4	5	
FCAIM	2.212	1.533	1.79	1.367	1.77	1.206	0.888	5.588	3.477	1.389	1.253	5.518	4.048	4.123	1.719	3.44	4.12
	(0.084)	(0.085)	(0.026)	(0.105)	(0.098)	(0.113)	(0.062)	(0.101)	(0.103)	(0.115)	(0.062)	(0.027)	(0.019)	(0.017)	(0.078)	(0.061)	
	7	4	3	4	4	4	3	4	6	2	4	6	4	2	4	5	
Khiops	2.539	2.901	3.353	3.093	2.938	4.421	4.324	6.141	3.583	2.938	4.867	6.925	6.353	5.504	2.805	4.853	9.69
	(0.091)	(0.154)	(0.069)	(0.058)	(0.1)	(0.078)	(0.049)	(0.05)	(0.045)	(0.078)	(0.052)	(0.016)	(0.025)	(0.041)	(0.054)	(0.049)	
	11	10	9	10	8	12	11	6	9	8	10	11	12	8	8	12	
MODL	1.853	1.642	1.939	1.498	1.761	0.926	1.001	5.003	2.721	2.337	1.503	5.503	4.407	4.295	1.73	3.273	4.81
	(0.103)	(0.069)	(0.102)	(0.063)	(0.106)	(0.112)	(0.079)	(0.093)	(0.102)	(0.088)	(0.098)	(0.016)	(0.031)	(0.029)	(0.039)	(0.064)	
	3	6	6	6	3	3	5	3	2	7	5	5	9	6	6	2	
CACC	2.342	2.053	1.922	2.134	2.134	1.251	1.287	7.976	3.729	1.912	1.169	5.614	4.089	4.224	2.559	3.434	6.88
	(0.151)	(0.197)	(0.254)	(0.131)	(0.28)	(0.255)	(0.401)	(0.114)	(0.229)	(0.103)	(0.047)	(0.012)	(0.02)	(0.126)	(0.288)	(0.062)	
	9	7	5	7	7	6	6	11	10	6	2	10	8	5	7	4	

a few generic conclusions can be drawn. CAIM, FCAIM, and MODL perform well for both quality measures, while Khiops, Maximum Entropy, and both unsupervised methods score relatively poorly. This observation is supported by the fact that the unsupervised methods are very simple and thus they cannot perform as well as the supervised algorithms. Similar observations were made in Kurgan and Cios (2004), but they concerned a smaller set of seven discretization methods. The tables show that CAIM, FCAIM, and MODL rank consistently well across all data sets, while the performance of other methods such as modified χ^2 and CADD varies between the data sets. Some discretizers, including CACC, IEM, and Patterson–Niblett, score very well for one criterion, while the quality of their schemes is shown to be poor when evaluated with the other measure.

The results concerning the average time of the discretization process, see Table 7, show (as expected) that the two unsupervised methods are the fastest. Although theoretically the Equal Width algorithm is faster than the Equal Frequency since it does not require sorting of feature values, we had to sort the values to compute the number of intervals d. As a result, both of the unsupervised methods have virtually identical running time. We note that if the user would specify the number of the intervals, the Equal Width algorithm, which is the only method that has linear complexity with respect to the number of feature values, would be the fastest.

The fastest supervised algorithm is FCAIM followed by CAIM and IEM. The slowest methods include MODL followed by Modified χ^2 and CADD. These numerical results stem from the design of these methods, that is, merging methods are in general slower than splitting methods. In addition, as shown in Table 8, the CAIM, FCAIM, and IEM methods generate a small number of intervals, which further improves their speed when compared with the other splitting methods. We observe that some differences are relatively large. For instance, for the page data set, which is characterized by the largest average number of distinct values, the unsupervised methods are two to three times faster than the fastest supervised methods, and they are three orders of magnitude faster than the slowest supervised methods. In the case of the pen data set, which has the largest number of instances, the fastest supervised and unsupervised discretizers are characterized by similar running time, while they are an order of magnitude faster than the slowest supervised methods.

In the next section, we aggregate the running time of the discretizers with the time required to build the classification model on the discretized data and the time to perform classification on the test fold to give further insights on the impact of discretization on the subsequent classification.

Table 8 gives the average number of intervals per continuous feature generated by the 12 algorithms. In general, schemes with a lower number of intervals cannot be considered better than schemes with a higher number because this information would have to be coupled with the information that quantifies how well these intervals describe the data distribution (e.g. CAIR or Entropy). However, if the distribution of the data is described equally well by two different discretization schemes, one should prefer the smaller scheme. We observe that the smallest schemes are generated by the IEM, FCAIM, CAIM, and MODL algorithms. For these algorithms, the number of intervals is often close to the number of classes in a given data set. We stress that the schemes generated by FCAIM, CAIM, and MODL are also characterized by favorable CAIR and entropy values. However, the biggest numbers of discretization intervals are generated by both unsupervised algorithms and by Maximum Entropy, Khiops, and Modified χ^2 algorithms. For unsupervised algorithms as well as for the Maximum Entropy, Paterson–Niblett, and CADD algorithms, the number of intervals is defined using the formula proposed in Wong and Chiu (1987); however, the supervised algorithms merge some of these intervals reducing the resulting number of discretization intervals. The Khiops and ME discretizers also obtain poor CAIR and entropy values, see Tables 5 and 6, which indicate that these schemes are characterized by relatively poor quality. Our results show that Modified χ^2 generates schemes with a large number of intervals when dealing with large data sets such as vowel, seg, and pen, which is consistent with findings in (Tay & Shen, 2002).

Overall, we conclude that the best trade-off between the running time, the quality of the discretization scheme expressed using CAIR and Entropy, and the number of discretization intervals is achieved by the FCAIM, CAIM, and IEM algorithms. The MODL method has a

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 7 The average running time (ms) associated with the discretization performed by the 12 discretization algorithms on the 16 benchmark data sets. The values in round brackets indicate standard deviation (over the 10-folds). The values in third row show rank of a given discretization algorithm for a given data set. Bolded values indicate the best results for a given data set. Last column shows average rank of a given algorithm scored over all data sets. Abbreviations for data set names can be found in Table 4

Algorithm	Anneal	Colic	Credit	Cylinder	Heart	Hypo	Sick	Vowel	Glass	Iono	Page	Pen	Sat	Seg	Sonar	Vehicle	Avg. rank
Equal Width	1.095	0.359	**0.784**	**1.514**	0.511	5.596	5.223	**1.818**	0.296	**2.137**	**12.28**	52.84	**45.75**	**7.64**	2.35	2.516	1.56
	(0.153)	(0.039)	**(0.11)**	**(0.202)**	(0.43)	(0.876)	(0.117)	**(0.184)**	(0.026)	**(0.36)**	**(1.563)**	(8.61)	**(7.455)**	**(0.055)**	(0.173)	(0.027)	
	2	2	**1**	**1**	2	2	2	**1**	2	**1**	**1**	2	**1**	**1**	2	2	
Equal frequency	**1.049**	**0.341**	0.817	1.554	**0.319**	**5.131**	**4.727**	2.056	**0.289**	2.196	13.05	**44.86**	48.23	7.687	**2.23**	**2.508**	1.44
	(0.166)	**(0.046)**	(0.143)	(0.123)	**(0.045)**	**(0.552)**	**(0.438)**	(0.025)	**(0.019)**	(0.376)	(2.151)	**(5.504)**	(6.334)	(0.061)	**(0.212)**	**(0.01)**	
	1	**1**	2	2	**1**	**1**	**1**	2	**1**	2	2	**1**	2	2	**1**	**1**	
Maximum Entropy	3.165	4.521	46.22	22.44	6.081	171	208	139	3.259	112	1488	454	368	909	178	33.88	6.06
	(0.833)	(1.063)	(22.02)	(1.758)	(2.188)	(53.52)	(45.74)	(6.373)	(0.757)	(16.36)	(264)	(77.8)	(65.71)	(114)	(11.07)	(3.04)	
	3	7	9	8	7	6	8	4	3	10	6	4	4	5	9	4	
Paterson Niblett	15.44	4.935	22.61	19.47	4.815	360	119	2220	27.08	79.59	6884	7347	2664	2701	79.79	92.8	8.5
	(1.946)	(0.692)	(2.52)	(1.979)	(0.759)	(50.63)	(11.49)	(13.18)	(2.875)	(15.24)	(1439)	(254)	(351)	(368)	(6.251)	(14.9)	
	10	8	6	6	6	9	6	10	10	9	10	10	10	10	7	9	
IEM	9.354	4.391	25	19.63	6.318	148	132	323	8.468	45.95	1376	2580	1970	1349	44.34	63.26	6.44
	(1.78)	(0.888)	(6.869)	(2.793)	(1.112)	(20.01)	(20.93)	(49.15)	(1.142)	(7.77)	(218)	(203)	(280)	(334)	(7.016)	(1.565)	
	7	5	7	7	8	5	7	6	4	6	5	7	8	7	6	8	
CADD	10.3	12.59	338	126	31.17	587	1037	609	8.96	1697	41550	555	581	11954	1105	195	9.94
	(2.181)	(2.431)	(41.61)	(18.09)	(5.986)	(89.47)	(319)	(54.98)	(3.071)	(381)	(5583)	(78.27)	(100)	(1196)	(283)	(26.93)	
	8	10	12	12	12	10	12	9	6	12	12	5	5	12	12	10	
Modified χ^2	35.52	28.4	102	85.79	16.92	744	774	192	10.33	181	3947	9193	11184	769	354	203	9.81
	(7.737)	(3.483)	(5.673)	(14.22)	(2.811)	(68.02)	(204)	(54.41)	(2.732)	(23.28)	(384)	(2645)	(4036)	(266)	(299)	(22.08)	
	11	12	10	10	11	11	10	5	8	11	9	11	12	4	11	11	
CAIM	8.652	2.862	15.08	11.93	2.61	202	99.95	479	11.78	29.84	2939	2114	1665	1740	31.95	46.58	5.88
	(1.235)	(0.399)	(2.341)	(1.683)	(0.634)	(17.95)	(8.333)	(80.9)	(2.917)	(4.004)	(431)	(11.9)	(286)	(291)	(4.427)	(8.239)	
	6	4	5	5	3	7	5	7	9	5	7	6	7	8	4	6	
FCAIM	6.008	2.071	10.4	9.593	3.122	74.14	46.79	481	8.984	12.4	849	2899	1231	1034	18.69	40.53	5
	(0.96)	(0.368)	(2.131)	(1.335)	(0.436)	(9.038)	(4.527)	(81.18)	(1.497)	(1.36)	(193)	(639)	(195)	(182)	(4.925)	(6.772)	
	5	3	4	4	5	4	4	8	7	3	4	9	6	6	3	5	
Khiops	3.801	4.486	7.31	6.113	2.997	14.09	12.49	61.29	8.69	27.31	114	91.1	89.39	152	39.06	9.987	3.62
	(0.435)	(0.67)	(6.999)	(0.864)	(1.646)	(2.275)	(1.338)	(5.635)	(5.976)	(2.216)	(24.99)	(55.03)	(23.84)	(102)	(28.34)	(0.972)	
	4	6	3	3	4	3	3	3	5	4	3	3	3	3	5	3	
MODL	39.17	12.66	123	86.34	10.55	983	911	3775	31.81	79.23	23035	12346	7011	11107	82.31	430	10.81
	(7.137)	(1.234)	(26.76)	(5.215)	(1.539)	(112)	(179)	(493)	(6.995)	(10.55)	(3674)	(1702)	(741)	(1404)	(9.096)	(74.73)	
	12	11	11	11	10	12	11	11	11	8	11	12	11	11	8	12	
CACC	11.19	5.648	30.9	48.5	9.631	266	362	18595	33.27	60.92	3425	2687	1998	1983	198	61.9	8.94
	(1.988)	(3.412)	(47.87)	(17.75)	(11.08)	(84.95)	(207)	(1049)	(10.75)	(9.318)	(818)	(494)	(22.33)	(348)	(71.65)	(2.369)	
	9	9	8	9	9	8	9	12	12	7	8	8	9	9	10	7	

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 8 The average number of intervals for a continuous feature associated with the discretization performed by the 12 discretization algorithms on the 16 benchmark data sets. The values in round brackets indicate standard deviation (over the 10-folds). The values in third row show rank of a given discretization algorithm for a given data set. Values in bold indicate the smallest number of intervals for a given data set. Last column shows average rank of a given algorithm scored over all data sets. Abbreviations for data set names can be found in Table 4

Algorithm	Anneal	Colic	Credit	Cylinder	Heart	Hypo	Sick	Vowel	Glass	Iono	Page	Pen	Sat	Seg	Sonar	Vehicle	Avg. rank
Equal width	6	6.714	21.17	7.856	8.617	11.29	22.11	14.03	7	20.73	44.82	10	7	22.63	16.76	6.683	9.38
	(0)	(0.178)	(0.401)	(0.184)	(0.158)	(0.202)	(0.361)	(0.067)	(0)	(0.795)	(0.349)	(0)	(0)	(0.186)	(0.099)	(0.053)	
	9	11	11	11	11	10	11	7	8	11	10	5	3	11	11	10	
Equal frequency	6	6.714	21.17	7.856	8.617	11.29	22.11	14.03	7	20.73	44.82	10	7	22.23	16.76	6.683	9.31
	(0)	(0.178)	(0.401)	(0.184)	(0.158)	(0.202)	(0.361)	(0.067)	(0)	(0.795)	(0.349)	(0)	(0)	(0.189)	(0.099)	(0.053)	
	9	11	11	11	11	10	11	7	8	11	10	5	3	10	11	10	
Maximum Entropy	1.933	5.586	19.12	7.15	8.383	10.7	20.74	14.03	3.467	19.21	44.48	3.969	4	22.13	16.76	5.994	7.62
	(0.274)	(0.238)	(0.343)	(0.126)	(0.112)	(0.196)	(0.392)	(0.067)	(0.155)	(1.194)	(0.333)	(0.033)	(0)	(0.189)	(0.097)	(0.049)	
	2	10	10	10	10	9	10	7	4	10	9	2	2	9	10	8	
Paterson Niblett	5.817	2.3	2	2.167	2.033	4.957	**1.286**	14.98	5	2.238	8.53	19.59	11.21	6.763	2.033	4.922	5.75
	(0.346)	(0.171)	(0)	(0.114)	(0.07)	(0.901)	**(0.165)**	(0.042)	(0)	(0.258)	(0.931)	(0.773)	(0.608)	(0.2)	(0.052)	(0.272)	
	7	6	2	6	4	7	**1**	10	6	3	6	10	10	3	5	6	
IEM	3.4	**1.129**	**1.717**	**1.078**	**1.567**	**2.157**	1.529	**3.03**	**1.989**	2.509	4.95	8.488	9.381	7.242	**1.222**	**3.017**	2.31
	(0.402)	**(0.081)**	**(0.209)**	**(0.065)**	**(0.086)**	**(0.171)**	(0.096)	**(0.177)**	**(0.269)**	(0.164)	(0.314)	(0.21)	(0.252)	(0.197)	**(0.08)**	**(0.126)**	
	4	**1**	**1**	**1**	**1**	**1**	2	**1**	**1**	4	2	3	9	4	**1**	**1**	
CADD	**1.833**	5.357	17.82	6.572	7.567	3.8	15.71	13.99	3.367	14.75	29.15	**3.938**	**3.853**	19.41	16.16	5.811	6.44
	(0.272)	(0.236)	(0.474)	(0.18)	(0.335)	(0.5)	(0.628)	(0.074)	(0.166)	(0.462)	(1.115)	**(0.059)**	**(0.049)**	(0.172)	(0.181)	(0.137)	
	1	9	9	9	9	5	9	6	3	9	8	**1**	**1**	8	9	7	
Modified χ^2	14.88	2.214	5.15	1.178	2.9	4.329	7.271	383	44.74	2.674	48.48	98.12	38.94	295	1.267	6.656	8.25
	(0.599)	(1.311)	(2.636)	(0.101)	(1.019)	(1.167)	(3.889)	(2.546)	(1.501)	(0.829)	(12.72)	(0.17)	(28.31)	(2.148)	(0.064)	(0.831)	
	12	4	7	2	6	6	7	12	12	5	12	12	12	12	2	9	
CAIM	5.983	2	2	1.978	2	3.429	1.714	11	7.011	**1.912**	5	10	7	**6.132**	2	4	3.44
	(0.053)	(0)	(0)	(0.029)	(0)	(0)	(0)	(0)	(0.035)	**(0)**	(0)	(0)	(0)	**(0.028)**	(0)	(0)	
	8	2	2	3	2	3	3	4	10	**1**	3	5	3	**1**	3	2	
FCAIM	5.017	2	2	1.978	2	3.429	1.714	11	6.878	**1.912**	5	10	7	**6.132**	2	4	3.12
	(0.214)	(0)	(0)	(0.029)	(0)	(0)	(0)	(0)	(0.152)	**(0)**	(0)	(0)	(0)	**(0.028)**	(0)	(0)	
	6	2	2	3	2	3	3	4	7	**1**	3	5	3	**1**	3	2	
Khiops	4.65	4.657	6.567	5.55	4.817	15.83	15.57	9.96	3.956	4.926	23.77	28.45	29.39	15.04	4.115	9.644	7.88
	(0.183)	(0.443)	(0.326)	(0.26)	(0.364)	(0.663)	(0.495)	(0.363)	(0.107)	(0.208)	(0.76)	(0.444)	(0.414)	(0.517)	(0.126)	(0.246)	
	5	8	8	7	7	12	8	3	5	8	7	11	11	7	7	12	
MODL	3.233	2.257	2.617	2.033	2.183	2.243	1.986	5.73	2.944	4.144	**4.76**	8.875	8.717	7.289	2.235	4.006	4.31
	(0.344)	(0.113)	(0.261)	(0.054)	(0.214)	(0.136)	(0.125)	(0.279)	(0.307)	(0.134)	**(0.344)**	(0.121)	(0.176)	(0.211)	(0.063)	(0.142)	
	3	5	5	5	5	2	5	2	2	7	**1**	4	8	5	6	4	
CACC	6.95	4.1	4.033	6.322	5.333	5.743	5.871	306	15.42	3.244	5.05	10	7	9.784	11.19	4.289	7.12
	(0.981)	(1.321)	(5.964)	(1.127)	(3.599)	(2.569)	(4.124)	(15.72)	(4.301)	(0.278)	(0.053)	(0)	(0)	(5.852)	(3.615)	(0.11)	
	11	7	6	8	8	8	6	11	11	6	5	5	3	6	8	5	

drawback of being relatively slow, but it also generates high quality discretization schemes. However, the worst performing methods include the two unsupervised methods, as well as Maximum Entropy, and Khiops, although these methods are relatively fast.

3.2 Classification with Naïve Bayes and semi-Naïve Bayes algorithms

We compare 6 classifiers on the data discretized by the 12 discretization algorithms, and two classifiers on the raw data. Hence, we have generated 74 different setups (classifier and discretizer pairs). Each setup was run on all 16 data sets where for each data set we performed five twofold cross-validations, that is, a total of ten test folds for each data set was considered for each set. Owing to their excessive size, the tables that provide all results are available online at `http://biomine.ece.ualberta.ca/discretizationNB/`. The results are summarized using the Friedman *F*-test and Nemenyi test, as suggested in (Demšar, 2006). These tests allow one to compare the quality of classifiers (measured using accuracy) across multiple data sets and to show the corresponding results in a convenient graphical format.

First, we compare accuracy using the Friedman *F*-test (Iman & Davenport, 1980). This test checks the null hypothesis whether accuracies of the considered classifiers are statistically equal based on the average place scored by a classifier along all data sets. If the null hypothesis is rejected, then a *post hoc* test (the Nemenyi test (Nemenyi, 1963)) is performed. This test orders the classifiers according to their quality and allows the selection of subsets of classifiers that are statistically undistinguishable, that is, classifiers for which the accuracies are not statistically significantly different. The best performing classifier is the method which has the lowest rank; this method is located in the rightmost position on a Nemenyi test figure. The worst classifier has the highest rank and is located in the left far end of a Nemenyi test figure. The setups for which ranks differ by less than the CD value are statistically indistinguishable from one another for the considered set of classifier accuracies obtained on a given set of data sets. The Nemenyi test figures use the abbreviated names of the discretizers and classifiers that can be found in Tables 2 and 3, respectively.

Two sets of tests are performed. First, we compare the impact of the 12 discretization methods on the accuracy of the same classifier. We also compare these results against baseline setups in which NB and FNB are run on the raw data. Second, we perform the test when considering all 74 setups together. For all performed tests, the null hypothesis was rejected, and hence we perform the *post hoc* test and present the corresponding graphical representation.

The results for NB classifier, see Figure 1(a), show that the accuracy of this classifier can be significantly improved by using discretization. This confirms the results obtained in previous studies (Flores *et al.*, 2007; Lee, 2007). At the same time, for six discretization schemes, which were generated by the two unsupervised algorithms and the Maximum Entropy, CADD, Paterson–Niblett, and Khiops supervised algorithms, there is no statistical difference between the accuracies of the NB run on the raw data and on the discretized data. This means that in the context of the classification accuracy, usage of these discretizers would not benefit the user. However, a statistically significant difference is found for the remaining discretization algorithms including Modified χ^2, CAIM, IEM, FCAIM, CACC, and MODL and also when using the FNB classifier. The Nemenyi test also shows that the classification accuracies of classifiers generated with FNB, and classifiers generated using NB on the data discretized with Paterson–Niblett, Equal Frequency, Khiops, Modified χ^2, CAIM, IEM, FCAIM, CACC, and MODL are characterized by statistically insignificant differences. We observe that discretization with any of the considered 12 methods does not lead to statistically significant improvement when compared with results obtained with FNB. The five best performing discretizers for the NB classifier, that is, the methods that achieve the lowest ranks, are MODL, CACC, IEM, and FCAIM *ex aequo*, and CAIM.

LBR is a classifier which works only with discrete data. This method is applied on the data discretized by the 12 discretizers and is compared against the NB and FNB classifiers that use the raw data. The results for the LBR classifier, see Figure 1(b), show that the results obtained on the discretized data are better than the results obtained by both the NB and FNB classifiers on

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

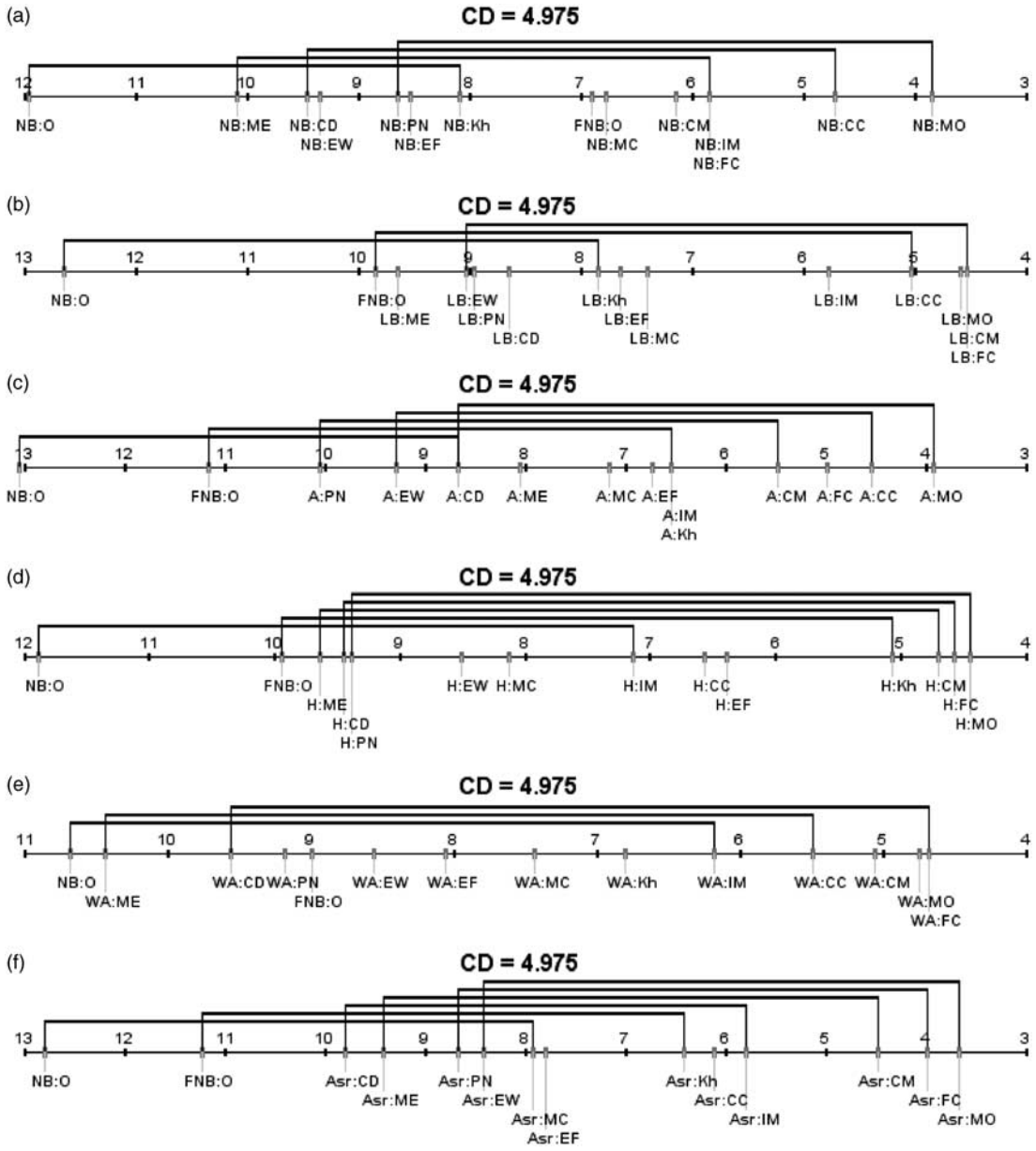


Figure 1 Graphical representation of the Nemenyi test performed for results obtained by NB and FNB run on the 16 original data sets. The results for the Naïve Bayes, LBR, AODE, HNB, WAODE, and AODEsr classifiers are shown in panels (a, b, c, d, e), and (f), respectively, and they concern runs on the 16 data sets discretized by the 12 discretization algorithms. The best result is obtained by the setup with the lowest rank; CD is the critical value which determines whether different setups are statistically different. The sets of classifiers that are not significantly different are shown using the horizontal lines. The considered setups are shown in the *x*:*y* form where *x* denotes a classifier and *y* denotes a discretization method; abbreviations for *x* and *y* can be found in Tables 2 and 3, respectively. O denotes the original (raw) data

the raw data. The results obtained for the data discretized by the five discretization algorithms (Maximum Entropy, Equal Width, Paterson–Niblett, CADD, and Khiops) are not statistically significantly better than the results obtained with NB. Statistically significant improvements over NB were obtained by using the remaining seven discretizers. The setups that are statistically significantly better than the results of both NB and FNB incorporate discretization with MODL, CAIM, and FCAIM. The five best results for the LBR classifier were obtained for schemes generated by the same discretization algorithms as the top five for the NB algorithm.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

For the AODE classifier, see Figure 1(c), all the results obtained on the discretized data are again better than the results obtained by NB and FNB. We observe that three setups, which concern the Paterson–Niblett, Equal Width, and CADD discretizers, generate classification models with accuracies that are not statistically significantly higher that the accuracies of the NB classifier on the raw data. The four setups that are significantly better than both NB and FNB use data discretized with MODL, CACC, FCAIM, and CAIM. These four setups are also included in the top five list for the NB and LBR classifier. The IEM algorithm is also in the top five list for the AODE classifier, but this time it occupies this spot *ex aequo* with the Khiops algorithm.

In the case of the HNB classifier, see Figure 1(d), we yet again observe that the setups which incorporate discretization perform better than the classification models generated by the NB and FNB classifiers on the raw data. This time, however, six setups are not significantly better than NB. The setup with the IEM algorithm, which is in the top five lists for all other classifiers, placed seventh. The setups that are significantly better than both NB and FNB applied on the raw data use data discretized with MODL, FCAIM, and CAIM. For the HNB classifier, the top five setups differ from the setup that scores best for others classifiers. More specifically, the IEM and CACC algorithms are replaced by the Equal Frequency and Khiops algorithms. However, the top three setups are consistent and they are based on the MODL, FCAIM, and CAIM algorithms.

Figure 1(e) presents results for the WAODE classifier. Only four setups are shown to be significantly better than the results obtained with the NB classifier. These setups include discretization performed with the FCAIM, MODL, CAIM, and CACC algorithms. We also observe that the results obtained by this classifier using the discretized data are not significantly better than the results obtained with FNB. There are three setups which obtained worse scores than FNB, and they are based on the Maximum Entropy, CADD, and Paterson–Niblett algorithms. The five best setups for WAODE are consistent with the top scoring setups for the NB, LBR, AODE, and AODEsr classifiers.

Results for the AODEsr classifier that are given in Figure 1(f) again show that NB and FNB executed on the raw data achieve the worst accuracies. Nevertheless, five setups are not significantly better than NB (CADD, Maximum Entropy, Paterson Niblett, Equal Width, and Modified χ^2) and the top five setups are statistically better than both NB and FNB. The latter group includes MODL, FCAIM, CAIM, IEM, and CACC, which is consistent with the top five lists of all other classifiers except HNB.

Overall, the results show that NB executed on the raw data is outperformed by NB and semi-NB classifiers that use discretized data. For all classifiers, except WODE and NB, the second worst rank was obtained by the FNB classifier, that is, discretization performed within FNB is outperformed by discretization performed by the considered 12 methods. The top five setups for all classifiers, except HNB, include the same five discretization algorithms, MODL, FCAIM, CAIM, IEM, and CACC. Moreover, the setups that incorporate these algorithms are always significantly better than the results obtained by using NB on the raw data; the only exception is the setup that includes IEM and WAODE. In the case of the HNB classifier, the best three setups include the MODL, FCAIM, and CAIM algorithms, and these discretizers are in the top three setups when considering other classifiers; except for NB where they are in the top five and AODE where CAIM is fourth. The results achieved by all setups that incorporate the unsupervised methods and older supervised methods, except IEM, were almost always comparable (not significantly better) with the results obtained by applying NB and FNB on the raw data. This shows that recently developed discretizers provide a significant improvement when compared with the older method. The setups with the two discretization methods based on χ^2 criterion (Modified χ^2 and Khiops) scored in the middle of the considered setups. At the same time, usage of the Khiops algorithm gave better results than setups that are based on the Modified χ^2 algorithm for all but the NB and LBR classifiers.

Figure 2 summarizes the results of the Nemenyi test for all considered setups. The large CD values are due to the large number of setups, that is, 74, when compared with the number of data sets, that is, 16. In spite of that, we observe that the results which are significantly better than the

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

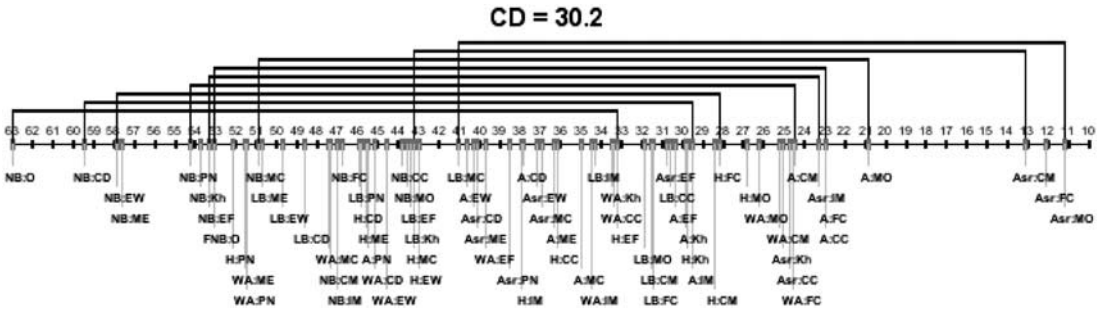


Figure 2 Graphical representation of the Nemenyi test performed for results obtained by NB and FNB run on the 16 original data sets, and the results for the 6 classifiers run on the 16 data sets discretized by the 12 discretization algorithms. The best result is obtained by the setup with the lowest rank; CD is the critical value which determines whether different setups are statistically different. The sets of classifiers that are not significantly different are shown using horizontal lines. The considered setups are shown in the *x*:*y* form where *x* denotes a classifier and *y* denotes a discretization method; abbreviations for *x* and *y* can be found in Tables 3 and 2, respectively. O denotes the original (raw) data

worst results that are obtained using NB on the raw data include almost all setups with the three discretization algorithms, MODL, FCAIM, and CAIM; only the setups with the NB classifiers are not in this group. However, all setups that use the NB classifier on the discretized data are shown to be comparable with the results obtained with the NB on the raw data. The setups which include the AODEsr and AODE classifiers (except the setup with the Paterson–Niblett discretizer) are among the best results. The best three setups which are on the far right end of the plot are setups that couple the AODEsr classifier with the MODL, FCAIM, and CAIM discretization algorithms. The big gap between these three and the remaining setups demonstrates a relatively big improvement between the scores of this group and the fourth and subsequent setups.

An alternative way to compare the accuracies is to count the wins, draws, and losses which a given setup achieves when compared against all other setups on all data sets (Demšar, 2006), that is, we compared each setup against every other setup on each data set. To compare whether a given setup is better than another setup, we used the combined 5×2 cv F-test (Alpaydin, 1999), that is, a win and a loss indicate that the difference is statistically significant. The table that gives all results can be found at `http://biomine.ece.ualberta.ca/discretizationNB/`. It compares all setups against the scores achieved by every classifier, that is, for each classifier, it sums up all wins, draws, and losses that a given setup achieves against all setups built using a given classifier. Table 9 summarizes the results by comparing a given setup with the aggregated (across all setups) results of another classifier. If the classifier is also used in the considered setup, then we can analyze how well a given discretization algorithm compares against all other discretization algorithms used with this classifier. We also compute a coefficient WDL_C

$$\mathrm{WDL}_C = \frac{W - L}{W + D + L} \tag{21}$$

where W, D, L stand for the number of wins, draws, and losses, respectively, and c indicates a given setup. If the coefficient value is bigger than 0, then the corresponding setup wins more often than it loses, whereas when the value is smaller than 0, the setup suffers more losses than wins. A value of 0 indicates an equal number of wins and losses. The coefficient values range between -1, that is, a given setup always loses, and 1, that is, a given setup always wins. The coefficients across all setups sum up to 0 since the number of wins and losses is equal. To better visualize the differences in Table 8, the cells, where WDL_C is bigger than 0.2, are given in bold. However, the cells that correspond to setups with WDL_C that is lower than -0.2 are underlined.

The setup that includes the NB classifier on the raw data is shown to often lose with other setups (it loses 553 times and wins only 68 times). Similarly, all setups with the NB classifier and

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 9 Number of wins, draws, and losses, respectively, achieved by a setup in a given row when compared against all setups (using all discretization methods) for a given classifier shown in the column. The setups include NB or FNB classifier run on the 16 raw data sets and each of the six classifiers run on the 16 data sets discretized by each of the 12 discretization algorithms. Bolded scores show WDL_C scores greater than 0.2; underlined scores correspond to WDL_C scores smaller than −0.2. Gray shading indicates comparison between setups that use the same classifier. The last column contains the aggregated, over all other setups, scores for a given setup. Abbreviations for algorithms and classifiers names can be found in Tables 2 and 3, respectively

	Original data		Discretized data						
	NB	FNB	NB	LBR	AODE	HNB	AODEsr	WAODE	Aggregated
Original Data									
NB	–	0/9/7	8/108/76	5/107/80	5/92/95	15/77/100	7/75/110	28/79/85	68/547/553
FNB	**7/9/0**	–	32/137/23	8/121/63	6/112/74	25/92/75	10/95/87	39/88/65	127/654/387
NB									
EW	**5/10/1**	1/11/4	18/138/20	5/115/72	3/108/81	19/83/90	5/81/106	28/85/79	84/631/453
EF	**8/8/0**	1/12/3	22/133/21	10/115/67	7/118/67	31/83/78	10/90/92	43/69/80	132/628/408
ME	3/12/1	1/12/3	9/106/61	7/96/89	5/86/101	27/70/95	9/75/108	40/64/88	101/521/546
PN	5/9/2	2/11/3	15/119/42	9/118/65	6/113/73	29/89/74	11/100/81	42/77/73	119/636/413
IM	**8/7/1**	3/11/2	33/134/9	13/122/57	13/116/63	36/84/72	15/98/79	45/75/72	166/647/355
CD	**5/10/1**	1/11/4	9/111/56	6/92/94	5/79/108	24/63/105	5/76/111	28/80/84	83/522/563
MC	**5/10/1**	2/11/3	25/122/29	6/129/57	9/111/72	29/90/73	11/96/85	38/86/68	125/655/388
CM	**8/8/0**	3/11/2	31/145/0	11/133/48	10/123/59	35/96/61	14/112/66	43/85/64	155/713/300
FC	**7/9/0**	2/12/2	33/142/1	11/133/48	11/123/58	35/93/64	15/107/70	44/81/67	158/700/310
Kh	**6/10/0**	1/13/2	17/124/35	6/102/84	5/88/99	24/75/93	7/73/112	31/73/88	97/558/513
MO	**8/8/0**	3/11/2	**42/134/0**	16/126/50	11/123/58	36/90/66	17/100/75	47/73/72	180/665/323
CC	**8/7/1**	3/11/2	30/136/10	9/132/51	10/119/63	33/90/69	14/108/70	44/81/67	151/684/333
LBR									
EW	**6/9/1**	**5/10/1**	52/126/14	16/133/27	13/109/70	32/100/60	14/98/80	42/86/64	180/671/317
EF	**8/8/0**	**4/12/0**	**68/123/1**	22/134/20	16/135/41	41/109/42	17/118/57	52/89/51	228/728/212
ME	**7/9/0**	**6/9/1**	**53/131/8**	13/126/37	9/129/54	31/101/60	10/103/79	43/89/60	172/697/299
PN	**7/8/1**	**5/10/1**	54/120/18	10/119/47	11/125/56	36/99/57	12/119/61	42/93/57	177/693/298
IM	**7/9/0**	**5/11/0**	**69/121/2**	35/135/6	25/138/29	46/98/48	30/106/56	57/84/51	274/702/192
CD	**5/10/1**	4/10/2	52/116/24	15/124/37	8/116/68	31/90/71	11/91/90	37/94/61	163/651/354
MC	**5/10/1**	4/10/2	47/123/22	9/132/35	14/129/49	30/114/48	17/119/56	43/96/53	169/733/266
CM	**8/8/0**	**7/9/0**	**87/105/0**	30/143/3	22/144/26	48/116/28	26/128/38	56/102/34	284/755/129
FC	**7/9/0**	**6/10/0**	**87/104/1**	30/141/5	23/143/26	48/116/28	28/127/37	56/101/35	285/751/132
Kh	**5/11/0**	3/13/0	**49/133/10**	17/124/35	14/127/51	34/99/59	16/103/73	43/93/56	181/703/284
MO	**7/9/0**	**6/10/0**	**86/105/1**	34/141/1	24/141/27	47/101/44	29/115/48	55/94/43	288/716/164
CC	**8/7/1**	**8/7/1**	**78/106/8**	31/136/9	25/138/29	48/109/35	27/120/45	55/95/42	280/718/170
AODE									
EW	**7/8/1**	4/11/1	**52/132/8**	39/128/25	17/126/33	42/103/47	16/111/65	46/99/47	223/718/227
EF	**9/6/1**	**8/8/0**	**85/105/2**	47/136/9	23/140/13	45/123/24	25/126/41	56/106/30	298/750/120
ME	**8/8/0**	**6/10/0**	**76/98/18**	38/128/26	18/112/46	37/105/50	17/114/61	49/98/45	249/673/246
PN	**7/8/1**	**5/10/1**	52/124/16	13/133/46	11/111/54	34/99/59	12/108/72	44/82/66	178/675/315
IM	**7/9/0**	**5/10/1**	**72/120/0**	**52/132/8**	31/133/12	57/113/22	27/130/35	57/105/30	308/752/108
CD	**7/8/1**	**6/8/2**	**79/92/21**	47/108/37	17/115/44	43/102/47	21/100/71	48/99/45	268/632/268
MC	**6/9/1**	4/11/1	53/117/22	17/142/33	15/125/36	37/112/43	18/126/48	48/103/41	198/745/225
CM	**9/7/0**	**8/8/0**	**91/101/0**	**61/130/1**	34/140/2	57/116/19	26/133/33	58/109/25	**344/744/80**
FC	**9/7/0**	**8/8/0**	**90/102/0**	**62/130/0**	33/142/1	**59/117/16**	27/138/27	**58/115/19**	**346/759/63**
Kh	**9/7/0**	**6/10/0**	**87/97/8**	41/140/11	23/125/28	41/121/30	18/132/42	54/106/32	279/738/151
MO	**9/7/0**	**7/9/0**	**88/104/0**	**61/131/0**	34/138/4	**56/123/13**	33/142/17	**61/115/16**	**349/769/50**
CC	**8/8/0**	**7/9/0**	**77/115/0**	**48/136/8**	29/135/12	55/120/17	25/143/24	58/108/26	307/774/87
HNB									
EW	**7/6/3**	4/9/3	63/96/33	47/106/39	34/107/51	27/120/29	17/115/60	32/129/31	231/688/249
EF	**10/5/1**	**7/7/2**	**91/65/36**	53/86/53	41/101/50	28/122/26	24/104/64	45/105/42	299/595/274
ME	**7/7/2**	**7/6/3**	85/59/48	44/88/60	23/98/71	20/101/55	14/97/81	40/105/47	240/561/367
PN	**6/9/1**	5/9/2	44/119/29	13/108/71	13/96/83	13/98/65	9/93/90	22/99/71	125/631/412
IM	**7/8/1**	**5/10/1**	**73/98/21**	59/108/25	35/126/31	30/126/20	26/124/42	36/134/22	271/734/163
CD	**7/6/3**	6/6/4	80/52/60	39/89/64	25/83/84	16/109/51	15/86/91	23/105/64	211/536/421
MC	**8/7/1**	5/8/3	62/89/41	20/112/60	14/113/65	19/117/40	17/115/60	34/112/46	179/673/316
CM	**11/5/0**	**8/7/1**	**98/82/12**	**68/112/12**	47/126/19	38/132/6	29/131/32	51/127/14	**350/722/96**
FC	**10/6/0**	**8/7/1**	**93/87/12**	**65/115/12**	45/126/21	40/130/6	28/131/33	51/127/14	**340/729/99**
Kh	**9/5/2**	**6/8/2**	**86/71/35**	54/101/37	36/112/44	36/111/29	26/108/58	48/121/23	301/637/230

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Table 9 *(Continued)*

MO	**9/7/0**	**7/8/1**	**86/94/12**	**67/111/14**	41/138/13	**43/131/2**	30/138/24	**52/134/6**	**335/761/72**
CC	**9/6/1**	**7/7/2**	**79/94/19**	51/116/25	33/128/31	32/131/13	25/126/41	46/121/25	282/729/157
AODEsr									
EW	**9/6/1**	**5/10/1**	**77/104/11**	55/118/19	44/121/27	55/112/25	22/126/28	56/115/21	323/712/133
EF	**10/5/1**	**9/6/1**	**94/79/19**	66/94/32	47/123/22	51/117/24	25/129/22	58/113/21	360/666/142
ME	**9/6/1**	**6/9/1**	**91/73/28**	51/101/40	33/110/49	42/103/47	14/101/61	49/101/42	295/604/269
PN	**9/7/0**	**6/9/1**	**64/123/5**	26/128/38	22/122/48	42/106/44	13/110/53	50/90/52	232/695/241
IM	**8/8/0**	**6/10/0**	**82/110/0**	**63/128/1**	**53/134/5**	**67/116/9**	34/131/11	**62/120/10**	**375/757/36**
CD	**7/8/1**	**7/7/2**	**81/87/24**	55/101/36	31/123/38	36/118/38	14/109/53	47/113/32	278/666/224
MC	**7/7/2**	**6/8/2**	**67/99/26**	19/134/39	20/131/41	32/118/42	15/118/43	47/105/40	213/720/235
CM	**11/5/0**	**9/7/0**	**104/87/1**	**83/107/2**	**57/135/0**	**74/117/1**	36/138/2	**70/122/0**	**444/718/6**
FC	**11/5/0**	**9/7/0**	**103/88/1**	**86/104/2**	**60/131/1**	**74/118/0**	**38/137/1**	**71/121/0**	**452/711/5**
Kh	**10/6/0**	**7/8/1**	**96/87/9**	**69/105/18**	50/120/22	55/118/19	27/119/30	**66/105/21**	**380/668/120**
MO	**10/6/0**	**8/8/0**	**101/91/0**	**81/111/0**	**70/122/0**	**81/110/1**	**46/129/1**	**72/120/0**	**469/697/2**
CC	**9/6/1**	**9/6/1**	**95/88/9**	**66/116/10**	49/131/12	**67/115/10**	36/125/15	**71/106/15**	**402/693/73**
WAODE									
EW	**9/4/3**	4/8/4	69/84/39	52/93/47	40/98/54	34/119/39	22/113/57	23/121/32	253/640/275
EF	**8/6/2**	**7/7/2**	76/68/48	52/85/55	39/95/58	29/122/41	19/112/61	24/130/22	254/625/289
ME	**8/5/3**	6/7/3	80/52/60	43/80/69	17/97/78	21/98/73	11/93/88	14/108/54	200/540/428
PN	**6/8/2**	4/8/4	51/101/40	15/106/71	13/100/79	18/109/65	9/86/97	11/106/59	127/624/417
IM	**6/9/1**	4/10/2	**71/92/29**	57/101/34	45/106/41	48/113/31	28/115/49	30/133/13	289/679/200
CD	6/7/3	5/7/4	**81/69/42**	44/96/52	25/111/56	19/128/45	12/112/68	16/116/44	208/646/314
MC	4/9/3	4/7/5	55/77/60	11/109/72	13/104/75	13/118/61	7/105/80	18/118/40	125/647/396
CM	**8/6/2**	**7/6/3**	**87/74/31**	**72/88/32**	51/108/33	47/129/16	30/127/35	33/142/1	335/680/153
FC	**8/6/2**	**7/6/3**	**87/74/31**	**73/87/32**	49/110/33	48/128/16	30/128/34	34/141/1	336/680/152
Kh	**8/4/4**	6/6/4	**85/71/36**	58/94/40	38/104/50	35/118/39	25/107/60	29/118/29	284/622/262
MO	**7/8/1**	**6/8/2**	**87/80/25**	71/87/34	53/104/35	51/119/22	36/112/44	**39/137/0**	350/655/163
CC	**7/7/2**	5/8/3	**73/87/32**	59/90/43	39/108/45	42/118/32	25/121/46	38/124/14	288/663/217

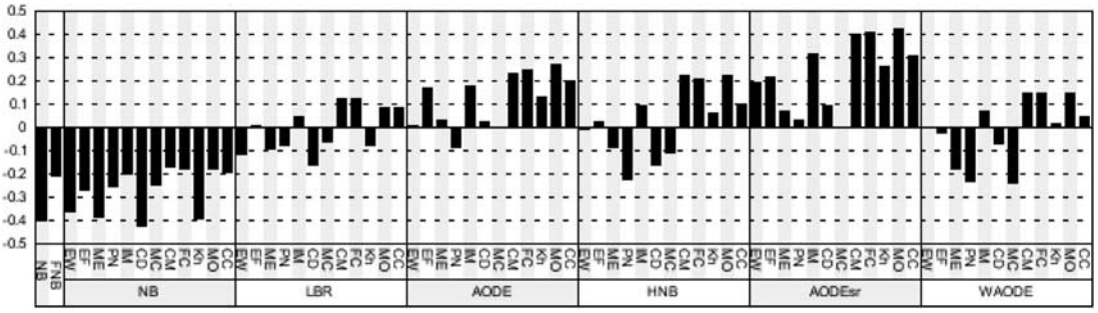


Figure 3 WDL_C values for all considered setups. The values are based on comparison with all other setups except the setups that include the same classifier. The setups include NB or FNB executed on the 16 raw data sets and each of the 6 classifiers on the 16 data sets discretized by each of the 12 algorithms. Abbreviations for algorithms and classifier names can be found in Tables 2 and 3, respectively

discretized data register more losses than wins. The two best classifiers are AODE and AODEsr. The setups that involve these classifiers are almost always characterized by more wins than losses.

The WDL_C values are visualized in Figures 3 and 4. The former figure gives the coefficient values when comparing a given setup against all other setups that use a different classifier, while the latter figure shows the values when comparing a given setup against all other setups that use the same classifier. As such, Figure 3 allows comparison between a given setup and other classifiers and Figure 4 enables comparison of different discretization algorithms for the same classifier.

Figure 3 shows that the best results along all considered classifiers were obtained by the AODEsr classifier for which all setups have a positive coefficient (the coefficient for the setup which includes a Modified χ^2 discretizer is positive and close to 0). On the opposite end, all

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

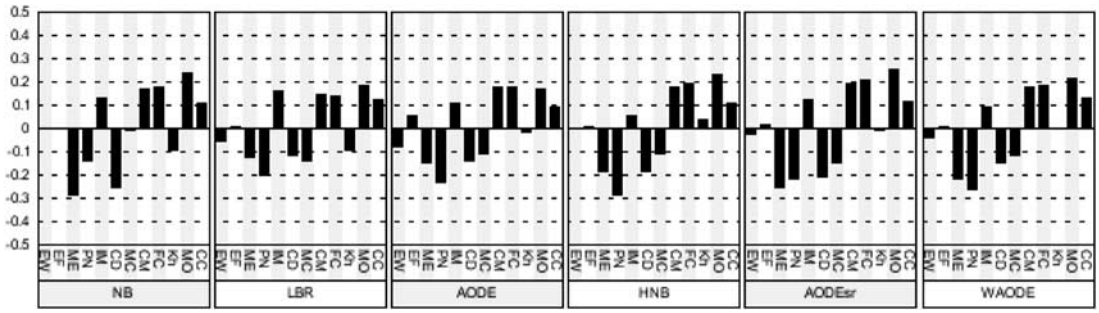


Figure 4 WDL_C values for all considered setups. The values are based on comparison among setups that are based on the same classifier. The setups include NB or FNB executed on the 16 raw data sets and each of the six classifiers on the 16 data sets discretized by each of the 12 algorithms. Abbreviations for algorithms and classifier names can be found in Tables 2 and 3, respectively

coefficient values for the NB classifier are negative. Relatively good scores were also obtained with the AODE and HNB classifiers.

Most importantly, we observe that the scores obtained by each classifier strongly depend on the discretization method used to discretize the input data. Figure 4 shows that five discretizers, which include MODL, FCAIM, CAIM, IEM, and CACC, generate the highest and always positive values across all considered classifiers. However, several algorithms such as Maximum Entropy, Paterson–Niblett, CADD, and Modified χ^2 always obtain negative coefficient values. The latter group incorporates relatively old algorithms, except for the Modified χ^2. The remaining algorithms have a similar number of losses and wins. The above conclusions are similar to the observation derived when using the Freidman F and Nemenyi tests.

3.3 Running time analysis

We also investigate the trade-off between the classification accuracy and the running time necessary to compute the discretization scheme and the classification model, and to compute predictions for the test instances. We note that the classification model is built only once, while it can be used to classify an unlimited number of test instances. This motivates us to consider the time to predict the test instances. In addition, we note that some of the semi-NB classifiers, such as LBR, are characterized by higher complexity associated with the prediction, see Table 3.

Similarly, as in the case of the classification model, the discretization process is performed only once before a classifier is built. After that, the generated discretization scheme can be used to discretize multiple samples. The time needed to discretize a single instance with a given scheme is directly proportional to the number of continuous attributes and the number of intervals in a given scheme. Therefore, the smaller the average number of intervals in a given scheme is, the faster the discretization process is. We observe that this time is negligible when compared to the time necessary to compute the scheme and the classifier, that is, the conversion into the discrete feature uses a pre-computed look up table that represents the discretization scheme, and thus we disregard it in our analysis.

Owing to the limited amount of space, our analysis is based on three data sets. We choose the smallest (*glass*), the average size (*hypo*), and the biggest (*pen*) data sets. The remaining plots are given on `http://biomine.ece.ualberta.ca/discretizationNB/`. The plots show the average time (over the five twofold cross-validation experiments) to build the discretization scheme using black bars, the average time to train a classifier using white bars, and the average time to classify a test set using gray bars. We focus on the analysis of the training and prediction times since the discretization time was already discussed in the context of Table 7.

Figure 5(a) summarizes the results for the smallest data set. It shows that the process to compute the discretization scheme can be time-consuming when compared with the time needed to

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

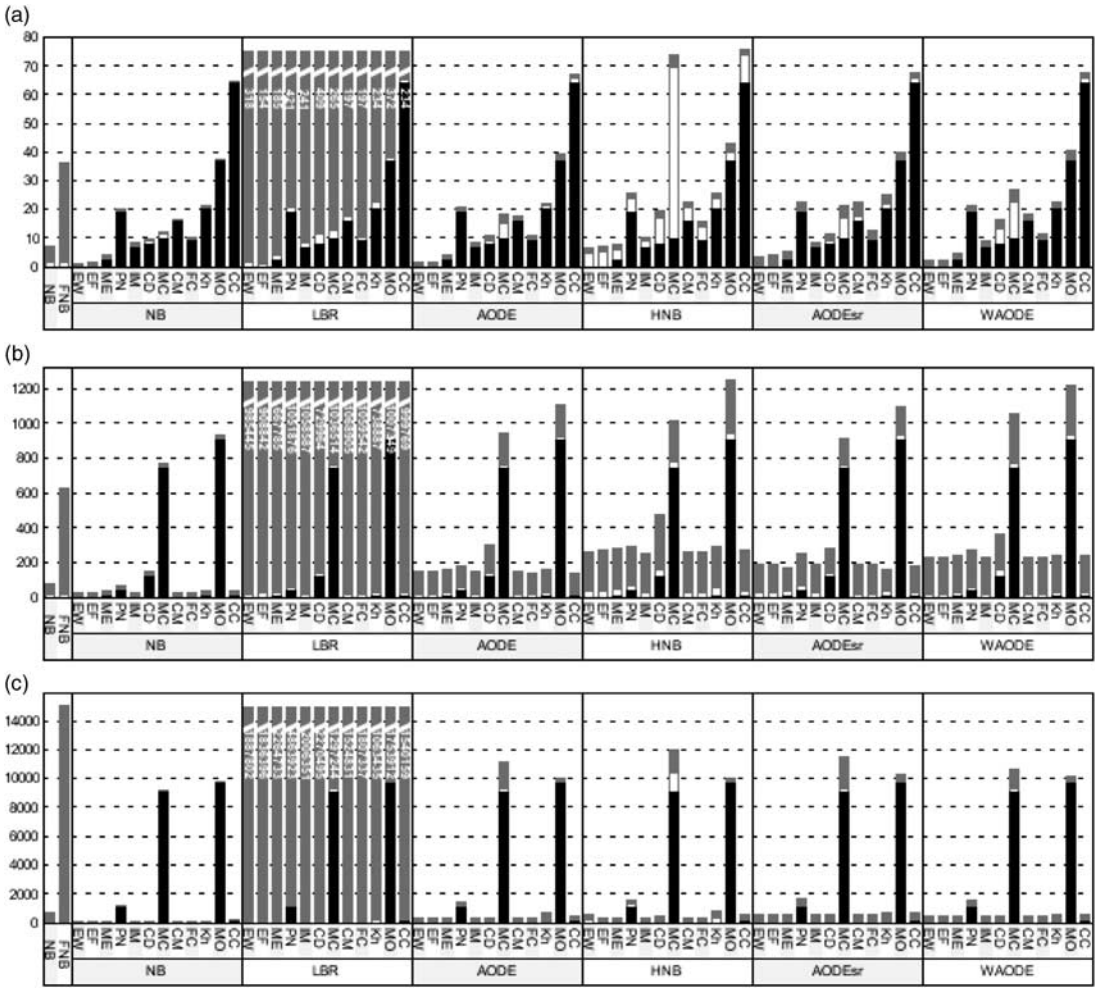


Figure 5 Time plots for the glass, hypo and pen data sets are shown in panels (a, b), and (c), respectively. The *y* axis gives the time in milliseconds; the *x* axis lists all setups including NB and FNB executed on the raw data and the six classifiers run on the data discretized by each of the 12 algorithms. Black bars show the time of the discretization process on the training fold; white color represents the time to train the classification model on the training fold; gray color denotes the time needed to classify the test fold. Training and test folds have the same number of instances. Abbreviations for algorithms and classifier names can be found in Tables 2 and 3, respectively

build the classification model and to classify the test instances. However, the time needed to classify discretized instances by any classifier, except LBR, is generally lower than the time needed to classify the original instances, which is associated with the NB and FNB classifiers. Most importantly, we observe that the whole process of discretizing, training, and classifying data is almost always shorter, except for the setups that apply the CACC discretizer and the LBR classifier, than the process of training and classifying the raw data using the FNB classifier. This is particularly important in the context of the fact that the building of the discretization scheme and the classification model is done only once, whereas the time needed to classify test instances is additive.

The longest training time is required for the HNB classifier and it is also associated with the setups in which data were discretized with the Modified χ^2 algorithm. The latter is likely caused by the relatively large number of intervals generated for this particular data set by the Modified χ^2 method, see Table 8. The setups which incorporate the CACC and CADD discretizers need more time to build the classification model. Although the schemes generated by CACC include a relatively large number of intervals, this is not the case for the CADD algorithm. Therefore, we hypothesize that the reason for the long training time is poor quality of the corresponding schemes.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Figure 5(b) gives the results for the hypothyroid data sets. This data set has average size and it consists of seven continuous and 22 nominal features. Owing to the relatively small number of continuous features, the results show that the time to classify the raw instances by NB is shorter than the time necessary to generate classification with most of the discretization-based setups. Only the setups that use the NB classifier on the discretized data are faster than NB on the raw data. We again observe that the time to classify raw data by FNB is longer than the time needed to classify instances by most of the other classifiers, except LBR, that use the discretized data. The difference is not as large as for the glass data set since FNB is very efficient for nominal features that are present in the raw data set. The longest training time, that is, time to compute the classification model, is observed for HNB.

The results for the biggest data set are presented in Figure 5(c). This data set consists exclusively of continuous features. The observations are similar to the observations on the glass data set. The time to classify discretized data is generally lower than the time needed to classify the raw data by NB and FNB (except for the LBR classifier). This is especially transparent in the case of FNB that requires a substantial amount of time to predict the test instances. Most of the discretization times, except for the Paterson–Niblett, Modified χ^2, MODL, and CACC algorithms, are 5 to 10 times smaller than the time to classify the test set. The large number of intervals generated by the Modified χ^2 algorithm results in long training and classification times.

Similar conclusions can be drawn for the remaining data sets; see `http://biomine.ece.ualberta.ca/discretizationNB/` for the time plots. The longest time to classify the test instances is observed for the LBR classifier. This time is almost always two orders of magnitude greater than the combined time to perform discretization and generation of the classifier. Discretization reduces the time to build the classifier and to predict the test instances. This is especially transparent for data sets that consist only of numerical (continuous) features, while for a few smaller mixed (both continuous and nominal features) data sets the benefits are not always as significant. We observe that the ratio between the discretization and classification times is smaller for bigger data sets. Classification with FNB most often takes more time than combined discretization and classification with semi-NB classifiers. For larger data sets that consist of continuous features, the usage of the discretization-classification combo, except when slow discretization methods such as MODL, Modified χ^2, and Paterson–Niblett are used, is also faster than the NB classifiers executed on the raw data.

4 Summary and conclusions

The comparison of the 12 considered discretizers indicates that unsupervised methods are faster than supervised methods. The fastest supervised algorithm is FCAIM followed by CAIM and IEM. The slowest is MODL followed by Modified χ^2 and CADD. The best overall quality measured over multiple dimensions, which include running time, quality of the discretization scheme expressed using CAIR and Entropy criteria, and the number of discretization intervals, is achieved by the FCAIM, CAIM, and IEM algorithms. Three algorithms, MODL, FCAIM, and CAIM, generate the smallest discretization schemes and achieve favorable CAIR and entropy values. The MODL method has a drawback of being relatively slow, but it also generates high-quality discretization schemes. The IEM algorithm is fast and generates small schemes with good Entropy and acceptable CAIR values. The worst performing methods include the two unsupervised methods, Equal Width and Equal Frequency, and the ME and Khiops algorithms. We also note that the Modified χ^2 algorithm generates a large number of discretization intervals.

Analysis of the impact of discretization on the classification with NB and semi-NB classifiers shows that discretization generally improves the accuracy when compared against NB and FNB run on the raw data. At the same time, we observe that the choice of the discretization algorithm impacts the significance of the improvement. The MODL, FCAIM, and CAIM methods have the biggest positive influence on the accuracy of the subsequent classification. Their application in tandem with most of the semi-NB classifiers results in statistically significant improvements.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

The IEM and CACC algorithms also guarantee a good improvement, which is slightly better than the improvement obtained with the Khiops algorithm. The unsupervised methods, as well as the Modified χ^2, Paterson–Niblett, CADD, and ME algorithms, produce relatively large discretization schemes, and provide the smallest improvements for the subsequent classification.

When analyzing the performance of the considered classifiers, we found that the most accurate models are generated by the AODEsr method followed by AODE and HNB. The classification accuracies obtained when combining AODEsr with the top performing discretizers that include MODL, FCAIM, and CAIM give statistically significant improvement over the classification performed with both classical NB methods, NB and FNB. The worst accuracies are obtained when using the original NB followed by FNB and LBR.

We found that the results of the evaluation of the discretization methods correlate with the accuracy of the classification performed on the discretized data, that is, well scoring discretizers provide good quality data, which in turn results in improved accuracies (irrespective of the type of classifier used). This shows that the quality indices used to evaluate discretization schemes can be used to indicate the quality of the classification with the NB-based classifiers.

Investigation into the impact of discretization on the classification time shows that although the time needed to build the discretization scheme could be longer than the time needed to train the classifier, the time to classify the discretized instances (that includes the time to discretize data, compute the classifier, and predict the test instances) is often shorter than the time needed to classify continuous instances. We show that the time needed to classify test instances is an important factor that is strongly influenced by discretization. At the same time, the time needed to build the classification model is not as strongly influenced by discretization.

We show that discretization not only has a positive impact on the accuracy of NB and semi-NB models, but it also has a positive impact on their classification time, that is, a strong positive influence is observed in the context of the time needed to classify an instance. The magnitude of this influence varies depending on which method was used to derive the discretization scheme. We found that the biggest positive influence, both on the accuracy and the classification time, is associated with three algorithms that generate the smallest discretization schemes, MODL, FCAIM, and CAIM.

Acknowledgements

This work was supported in part by the NSERC Discovery grant to L. Kurgan and by the Killam Memorial Scholarship to M. J. Mizianty.

References

Abraham, R., Simha, J. B. & Iyengar, S. S. 2006. A comparative analysis of discretization methods for medical datamining with naïve bayesian classifier. In *ICIT '06: Proceedings of the 9th International Conference on Information Technology*. IEEE Computer Society, Washington, DC, USA, 235–236.

Alpaydin, E. 1999. Combined 5×2 cv f test for comparing supervised classification learning algorithms. *Neural Computation* **11**(8), 1885–1892.

Asuncion, A. & Newman, D. 2007. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Boullé, M. 2004. Khiops: a statistical discretization method of continuous attributes. *Machine Learning* **55**(1), 53–69.

Boullé, M. 2006. MODL: a bayes optimal discretization method for continuous attributes. *Machine Learning* **65**(1), 131–165.

Catlett, J. 1991. On changing continuous attributes into ordered discrete attributes. In *EWSL '91: Proceedings of the European Working Session on Machine Learning*. Springer-Verlag, London, UK, 164–178.

Ching, J., Wong, A. & Chan, K. 1995. Class-dependent discretization for inductive learning from continuous and mixed mode data. *IEEE Transactions Pattern Analysis and Machine Intelligence* **17**(7), 641–651.

Cios, K. J. & Kurgan, L. A. 2002. *Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms*. Physica-Verlag GmbH.

Cios, K. J. & Kurgan, L. A. 2004. Clip4: hybrid inductive machine learning algorithm that generates inequality rules. *Information Sciences* **163**(1–3), 37–83.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Clark, P. & Niblett, T. 1989. The cn2 induction algorithm. *Machine Learning* **3**(4), 261–283.
Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* **7**, 1–30.
Dougherty, J., Kohavi, R. & Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *Proceedings of 12th International Conference Machine Learning*. Morgan Kaufmann, 194–202.
Fayyad, U. M. & Irani, K. B. 1992. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* **8**(1), 87–102.
Fayyad, U. M. & Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Uncertainty in AI*. Morgan Kaufmann, San Francisco, CA, USA, 1022–1027.
Flores, J. L., Inza, I. & Larran, P. 2007. Wrapper discretization by means of estimation of distribution algorithms. *Intelligent Data Analysis* **11**(5), 525–545.
Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P. & Smyth, P. 1997. Bayesian network classifiers. *Machine Learning* **29**, 131–163.
Huang, W. 1996. *Discretization of Continuous Attributes for Inductive Machine Learning*. MSc thesis. Department of Computer Science, University of Toledo, Ohio, USA.
Iman, R. L. & Davenport, J. M. 1980. Approximations of the critical region of the Friedman statistic. *Communications in Statistics* **A9**, 571–595.
Jiang, L. & Zhang, H. 2006. Weightily averaged one-dependence estimators. In *Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence*. Morgan Kaufmann, 970–974.
John, G. & Langley, P. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 338–345.
Kaufman, K. A. & Michalski, R. S. 1999. Learning from inconsistent and noisy data: the aq18 approach. In *Proceedings of the 11th International Symposium Methodologies for Intelligent Systems*, Saratoga Springs, NY, May 2005.
Keogh, E. J. & Pazzani, M. J. 1999. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of The Seventh International Workshop on Artificial Intelligence and Statistics*, Fort Lauderdale, Florida, USA, 3–6 January, 225–230.
Kerber, R. 1992. Chimerge: discretization of numeric attributes. In *Proceedings of the 9th International Conference of Artificial Intelligence*, Cambridge, UK, 20–22 February, 123–128.
Kohavi, R. & Sahami, M. 1996. Error-based and entropy-based discretization of continuous features. In *Proceedings of the 2nd International Conference Knowledge Discovery and Data Mining*, Portland, Oregon, USA, 2–4 August, 114–119.
Kujala, J. & Elomaa, T. 2007. Improved algorithms for univariate discretization of continuous features. In *PKDD 2007: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Warsaw, Poland, 17–21 September, 188–199.
Kurgan, L. A. & Cios, K. J. 2001. Discretization algorithm that uses class-attribute interdependence maximization. In *Proceedings of the 2001 International Conference on Artificial Intelligence*, Seattle, Washington, USA, 4–10 August, 980–987.
Kurgan, L. A. & Cios, K. J. 2003. Fast class-attribute interdependence maximization (CAIM) discretization algorithm. In *Proceeding of International Conference on Machine Learning and Applications*, Los Angeles, California, USA, 23–24 June, 30–36.
Kurgan, L. A. & Cios, K. J. 2004. CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering* **16**, 145–153.
Kurgan, L. A., Cios, K. J. & Dick, S. 2006. Highly scalable and robust rule learner: performance evaluation and comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **36**(1), 32–53.
Langley, P., Iba, W. & Thompson, K. 1992. An analysis of bayesian classifiers. In *Proceedings of the Tenth Conference on Artificial Intelligence*. MIT Press, 223–228.
Langley, P. & Sage, S. 1994. Induction of selective bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 399–406.
Lee, C.-H. 2007. A hellinger-based discretization method for numeric attributes in classification learning. *Knowledge-Based Systems* **20**(4), 419–425.
Liu, H. & Setiono, R. 1997. Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering* **9**, 642–645.
Liu, H., Hussain, F., Tan, C. & Dash, M. 2002. Discretization: an enabling technique. *Data Mining and Knowledge Discovery* **6**(4), 393–423.
Liu, X. & Wang, H. 2005. A discretization algorithm based on a heterogeneity criterion. *IEEE Transactions on Knowledge and Data Engineering* **17**(9), 1166–1173.
Mehta, S., Parthasarathy, S. & Yang, H. 2005. Toward unsupervised correlation preserving discretization. *IEEE Transactions on Knowledge and Data Engineering* **17**(9), 1174–1185.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press

Mizianty, M. J., Kurgan, L. A. & Ogiela, M. R. 2008. Comparative analysis of the impact of discretization on the classification with naïve bayes and semi-naïve bayes classifiers. In *ICMLA '08: Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, San Diego, California, USA, 11–13 December, 823–828.

Nemenyi, P. 1963. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University.

Paterson, A. & Niblett, T. 1987. *Acls Manual*. Intelligent Terminals, Ltd.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Rissanen, J. 1978. Modeling by shortest data description. *Automatica* **14**, 465–471.

Tay, F. E. H. & Shen, L. 2002. A modified chi2 algorithm for discretization. *IEEE Transactions on Knowledge and Data Engineering* **14**(3), 666–670.

Tsai, C.-J., Lee, C.-I. & Yang, W.-P. 2008. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences* **178**(3), 714–731.

Wang, K. & Liu, B. 1998. Concurrent discretization of multiple attributes. In *Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*, Singapore, 22–27 November, 250–259.

Wang, Z. & Webb, G. I. 2002. Comparison of lazy bayesian rule and tree-augmented bayesian learning. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, IEEE Computer Society. Washington, DC, USA, 490.

Webb, G. I., Boughton, J. R. & Wang, Z. 2005. Not so naïve bayes: aggregating one-dependence estimators. *Machine Learning* **58**, 5–24.

Winter, R. & Auerbach, K. 2004. *Contents under Pressure*. Intelligent Enterprise. `http://www.intelligententerprise.com/showArticle.jhtml;?articleID=18902161`.

Witten, I. H. & Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann Publishers Inc.

Wong, A. K. C. & Chiu, D. K. Y. 1987. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9**(6), 796–805.

Wong, A. K. C. & Liu, T. S. 1975. Typicality, diversity, and feature pattern of an ensemble. *IEEE Transactions on Computers* **24**(2), 158–181.

Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J. & Steinberg, D. 2007. Top 10 algorithms in data mining. *Knowledge and Information Systems* **14**(1), 1–37.

Yang, Y. & Webb, G. I. 2002. A comparative study of discretization methods for naive-bayes classifiers. In *Proceedings of the 2002 Pacific Rim Knowledge Acquisition Workshop*, Tokyo, Japan, 18–19 August, 159–173.

Zhang, H., Jiang, L. & Su, J. 2005. Hidden naive bayes. In *Proceedings of the 20th National Conference on Artificial intelligence*. AAAI Press, 919–924.

Zheng, Z. & Webb, G. I. 2000. Lazy learning of bayesian rules. *Machine Learning* **41**(1), 53–84.

Zheng, F. & Webb, G. I. 2006. Efficient lazy elimination for averaged one-dependence estimators. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 1113–1120.

https://doi.org/10.1017/S0269888910000329 Published online by Cambridge University Press