Data Discretization

Hadi M.abachi

Faculty of computer science , Iran university of science & technology

Why discretization?

- Discretization of numerical data is one of the most influential data preprocessing tasks in knowledge discovery and data mining.
- Discretization is considered a data reduction mechanism because it diminishes data from a large domain of numeric values to a subset of categorical values.
- There is a necessity to use discretized data by many DM algorithms which can only deal with discrete attributes.
- Discretization causes that the learning methods show remarkable improvements in learning speed and accuracy. Besides, some decision tree-based algorithms produce shorter, more compact, and accurate results when using discrete values.
- Even with algorithms that are able to deal with continuous data, learning is less efficient and effective

➢ Nevertheless, any discretization process generally leads to a loss of information, making the minimization of such information loss is the main goal of a discretizer.

Discretization Process

- In supervised learning, and specifically in classification, the problem of discretization can be defined as follows. Assuming a dataset S consisting of N examples, M attributes, and c class labels, a discretization scheme.
- D_A would exist on the continuous attribute $A \in M$, which partitions this attribute into k discrete and disjoint intervals: $\{[d_0, d_1], (d_1, d_2], \ldots, (d_{k_A-1}, d_{k_A}]\}$,
- where d_0 and d_{kA} are, respectively, the minimum and maximal value, and represents the set of cut points of A in ascending order. $P_A = \{d_1, d_2, \ldots, d_{k_A-1}\}$

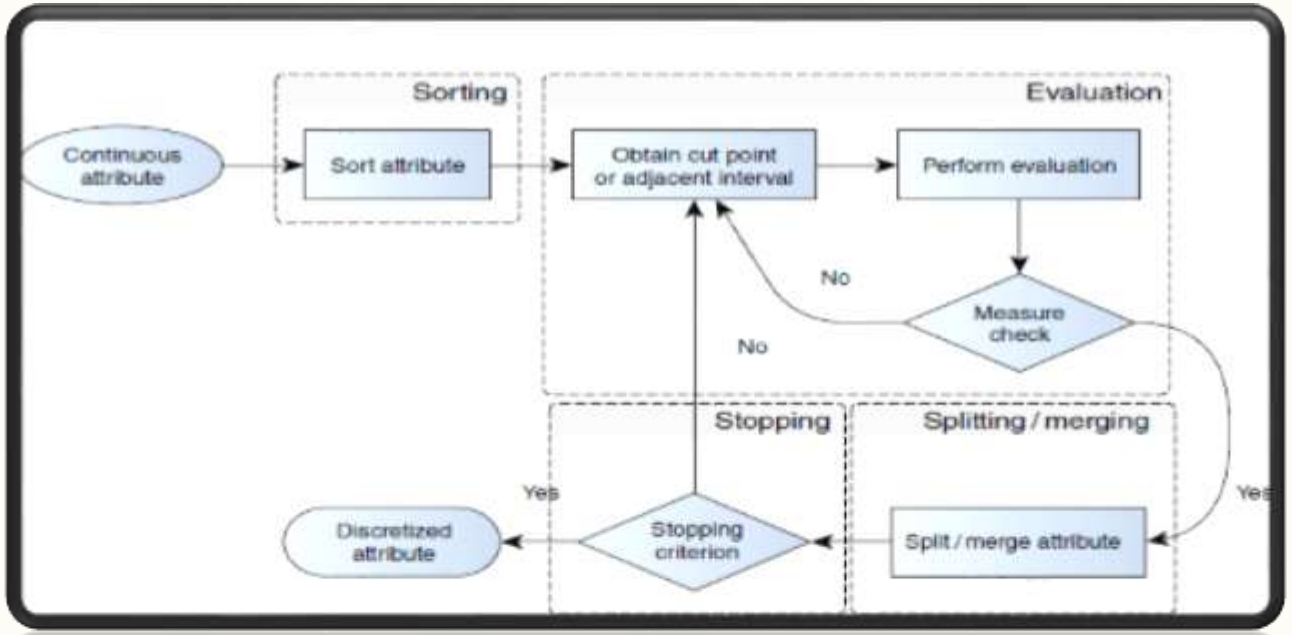
- A typical discretization process generally consists of four steps :

  (1) *sorting* the continuous values of the feature to be discretized,

  (2) *evaluating* a cut point for splitting or adjacent intervals for merging,

  (3) *splitting or merging intervals* of continuous values according to some *defined criterion.*

  (4) *stopping* at some point.



- Sorting: The continuous values for a feature are sorted in either descending or ascending order. It is crucial to use an efficient sorting algorithm with a time complexity of O(N logN). Sorting must be done only once and for the entire initial process of discretization. It is a mandatory treatment and can be applied when the complete instance space is used for discretization.

- Selection of a Cut Point: After sorting, the best cut point or the best pair of adjacent intervals should be found in the attribute range in order to split or merge in a following required step. An evaluation measure or function is used to determine the correlation, gain, improvement in performance, or any other benefit according to the class label.

- Splitting/Merging: Depending on the operation method of the discretizes, intervals either can be split or merged. For splitting, the possible cut points are the different real values present in an attribute. For merging, the discretizer aims to find the best adjacent intervals to merge in each iteration.

- Stopping Criteria: It specifies when to stop the discretization process. It should assume a tradeoff between a final lower number of intervals, good comprehension, and consistency.

Discretization Properties

• Static versus Dynamic: This property refers to the level of independence between the discretize and the learning method. A static discretize is run prior to the learning task and is autonomous from the learning algorithm, as a data preprocessing algorithm. By contrast dynamic discretizer responds when the learner requires so, during the building of the model.

• *Univariate versus Multivariate:* Univariate discretizersonly operate with a single attribute simultaneously. This means that they sort the attributes independently, and then, the derived discretization disposal for each attribute remains unchanged in the following phases. Conversely, multivariate techniques, concurrently consider all or various attributes to determine the initial set of cut points or to make a decision about the best cut point chosen as a whole. They may accomplish discretization handling the complex interactions among several attributes to decide also the attribute in which the next cut point will be split or merged.

• *Supervised versus Unsupervised:* Supervised discretizers consider the class label whereas unsupervised ones do not. The interaction between the input attributes and the class output and the measures used to make decisions on the best cutpoints (entropies, corrélations, etc.) will définie the supervised manner to discretize. Although most of the discretizers proposed are supervised, there is a growing interest in unsupervised discretization for descriptive tasks. Unsupervised discretization can be applied to both supervised and unsupervised learning, because its operation does not require the specification of an output attribute.

- Splitting versus Merging: These two options refer to the approach used to define or generate new intervals. The former methods search for a cut point to divide the domain into two intervals among all the possible boundary points. On the contrary, merging techniques begin with a predefined partition and search for a candidate cut point to mix both adjacent intervals after removing it.

- *Global versus Local:* In the time a discretizer must select a candidate cut point to be either split or merged, it could consider either all available information in the attribute or only partial information. A local discretizer makes the partition decision based only on partial information. The dynamic discretizers search for the best cut point during internal operations of a certain DM algorithm, thus it is impossible to examine the complete dataset.

- *Direct versus Incremental:* For direct discretizers, the range associated with an interval must be divided into *k* intervals simultaneously, requiring an additional criterion to determine the value of *k*. One-step discretization methods and discretizers which select more than a single cut point at every step are included in this category.
However, incremental methods begin with a simple discretization and pass through an improvement
process, requiring an additional criterion to determine when it is the best moment to stop. At each step, they find the best candidate boundary to be used as a cut point and, afterwards, the rest of the decisions are made accordingly.

Criteria to Compare Discretization Methods

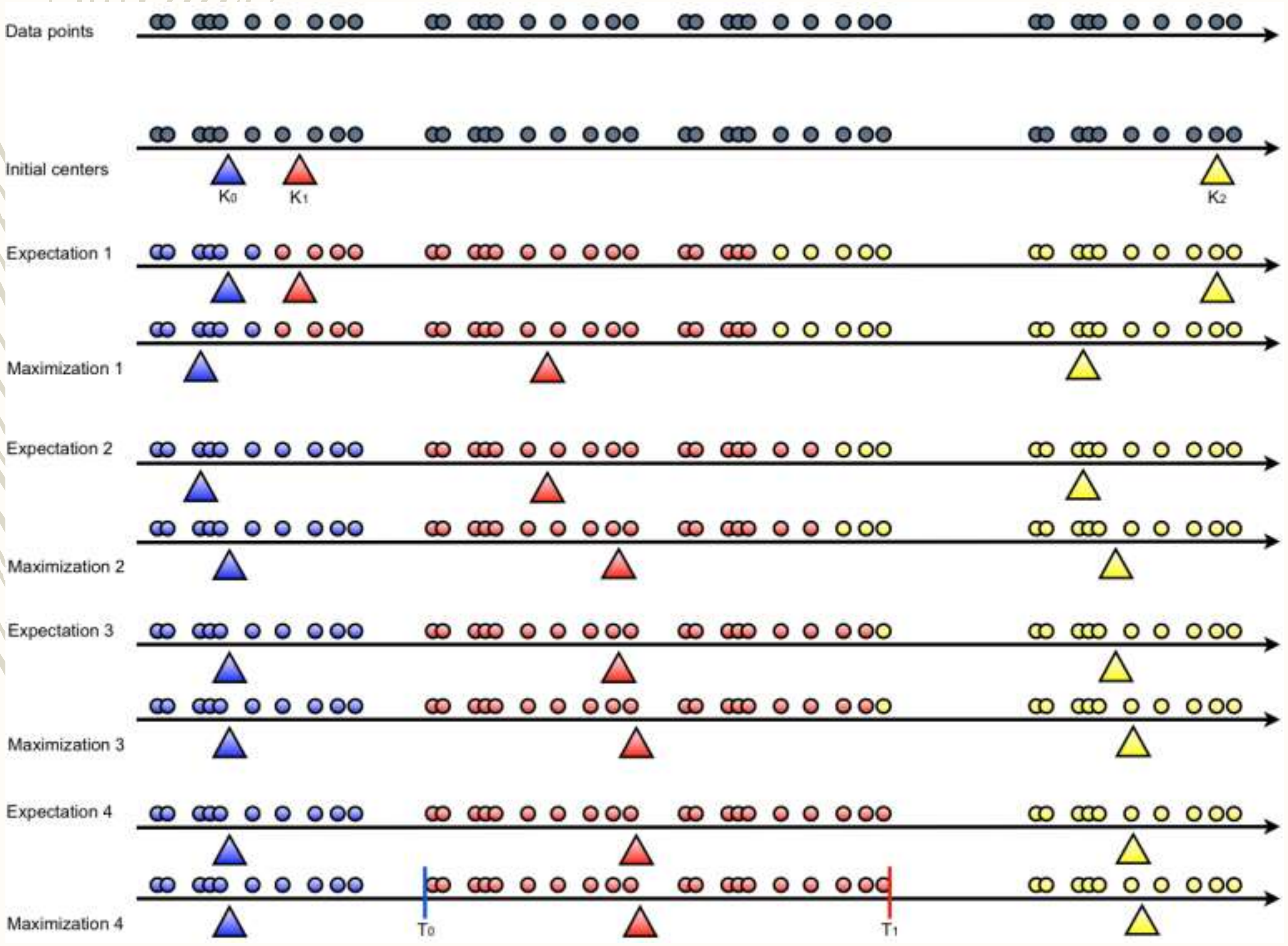
- ❑ Number of intervals: A desirable feature for practical discretization is that discretized attributes have as few values as possible, since a large number of intervals may make the learning slow and ineffective.
- ❑ Inconsistency: A supervision-based measure used to compute the number of unavoidable errors produced in the data set. An unavoidable error is one associated with two examples with the same values for input attributes and different class labels. In general, data sets with continuous attributes are consistent, but when a discretization scheme is applied over the data, an inconsistent data set may be obtained. The desired inconsistency level that a discretizer should obtain is 0.0.
- ❑ Predictive classification rate: A successful algorithm will often be able to discretize the training set without significantly reducing the prediction capability of learners in test data which are prepared to treat numerical data.
- ❑ Time requirements: A static discretization process is carried out just once on a training set, so it does not seem to be a very important evaluation method. However, if the discretization phase takes too long it can become impractical for real applications. In dynamic discretization, the operation is repeated many times as the learner requires, so it should be performed efficiently.

Discretization Methods and Taxonomy

Equal Width Discretizer	Multivariate Discretization
Equal Frequency Discretizer	Chi2
Chi Merge	LVQ based discretization
Minimum Description Length Principle	Genetic Algorithm discretizer
Fusinter	Rough set Discretizer

⋮

Examples: K-Means Discretization



This discretization algorithm is an unsupervised univariate discretization algorithm that:

1. Initialization: random creation of K centers.
2. Expectation: each point is associated with the closest center
3. Maximization: each center position is computed as the barycenter of its associated points

Minimum Description Length-Based Discretizer

This also introduces an optimization based on a reduction of whole set of candidate points, only formed by the boundary points in this set:
Let $A(e)$ denote the value for attribute A in the example e. Boundary point $b \in Dom(A)$ can be defined as the midpoint value between $A(u)$ and $A(v)$,assuming that in the sorted collection of points in A, *two* examples exist $u, v \in S$ with different class labels, such that $A(u) < b < A(v)$; and the other example $w \in S$ does not exist, such that $A(u) < A(w) < A(v)$. The set of boundary points for attribute A is defined as B_A.
It recursively evaluates all boundary points, computing the class entropy of the partitions derived
as quality measure. The objective is to minimize this measure to obtain the best cut decision.
Let b_a be a boundary point to evaluate, $S1 \subset S$ be a subset where$\forall\ a' \in S1$, $A(a') \leq b_a$ and $S2$ be equal to $S - S1$.
The class information entropy yielded by a given binary partitioning can be expressed as:

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

Where: E represents the class entropy of a given subset following Shannon's definitions:

$$Ent\left(S\right) = -\sum_{i=1}^{k} P(C_i, S)\log_2(P(C_i, S))$$

Finally, a decision criterion is defined in order to control when to stop the partitioning process. The use of MDLP as a decision criterion allows us to decide whether or not to partition. Thus a cut point b_a will be applied iff:

$$G(A, b_a, S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A, b_a, S)}{N},$$

where $\Delta(A, b_a, S) = \log_2(3^c) - [c\,E(S) - c_1E(S_1) - c_2E(S_2)]$, c_1 and c_2 the number of class labels in S_1 and S_2, respectively; and $G(A, b_a, S) = E(S) - EP(A, b_a, S)$.

BIG DATA BACKGROUND

The ever-growing generation of data on the Internet is leading us to managing huge collections using data analytics solutions. Exceptional paradigms and algorithms are thus needed to efficiently process these datasets so as to obtain valuable information, making this problem one of the most challenging tasks in Big Data analytics.

- Volume: the massive amount of data that is produced every day is still exponentially growing (from terabytes to Exabyte);
- Velocity: data need to be loaded, analyzed, and stored as quickly as possible.
- Veracity: the quality of data to process is also an important factor.
- Variety: data come in many formats and representations.

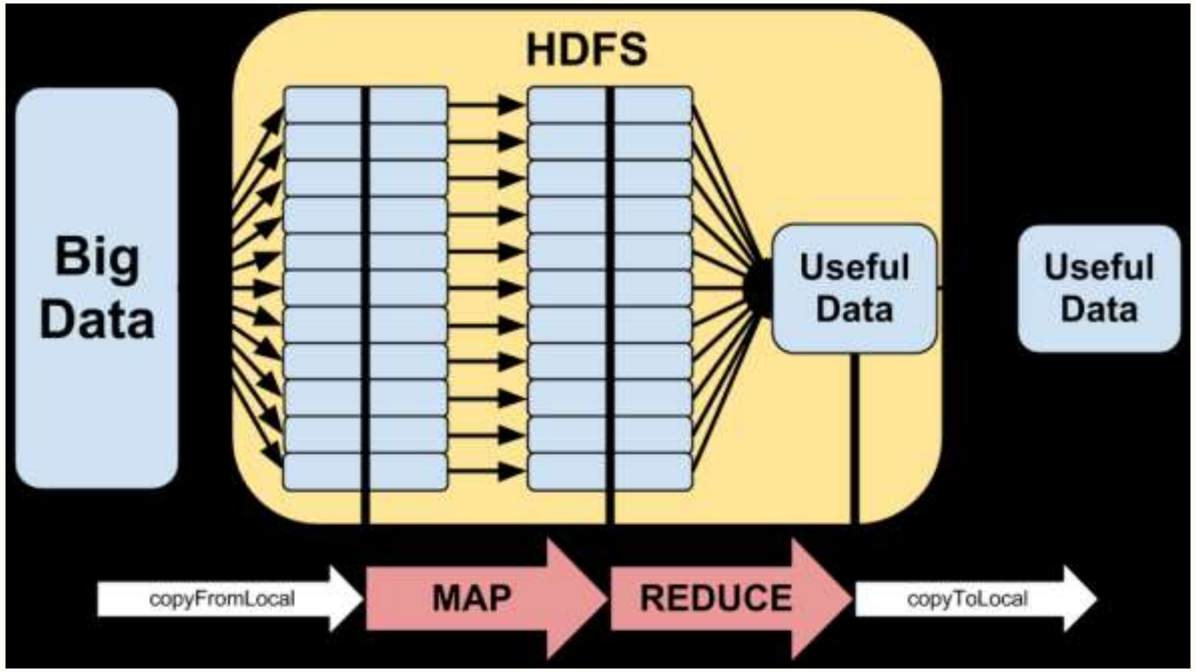
Question: Unsuitability of many knowledge extraction algorithms in the Big Data?

Solution: new methods to be developed to manage such amounts of data effectively and at a pace that allows value to be extracted from them.

Map Reduce Model and Other Distributed Frameworks

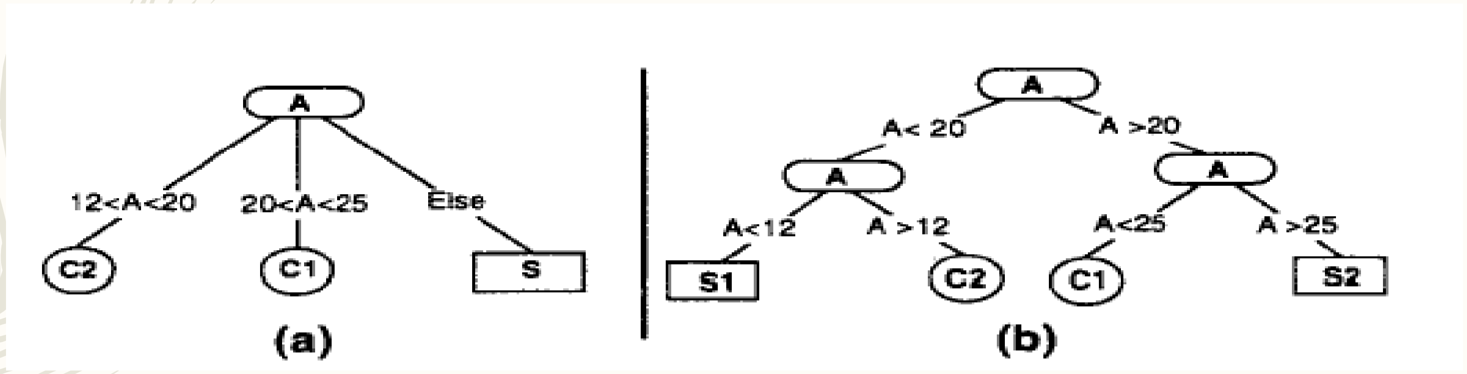
The Map Reduce framework, designed by Google in 2003, is currently one of the most relevant tools in Big Data analytics.

the master node breaks up the dataset into several splits, distributing them across the cluster for parallel processing. Each node then hosts several Map threads that transform the generated key-value pairs into a set of intermediate pairs.
After all Map tasks have finished, the master node distributes the matching pairs across the nodes according to a key-based partitioning scheme. Then the Reduce phase starts, combining those coincident pairs so as to form the final output.



DISTRIBUTED MDLP DISCRETIZATION

MDLP→ multi-interval extraction of points and the use of boundary points can improve the discretization process, both in efficiency and error rate.



Main Discretization Procedure: The algorithm calculates the minimum-entropy cut points by feature according to the MDLP criterion:

$$G(A, b_a, S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A, b_a, S)}{N},$$

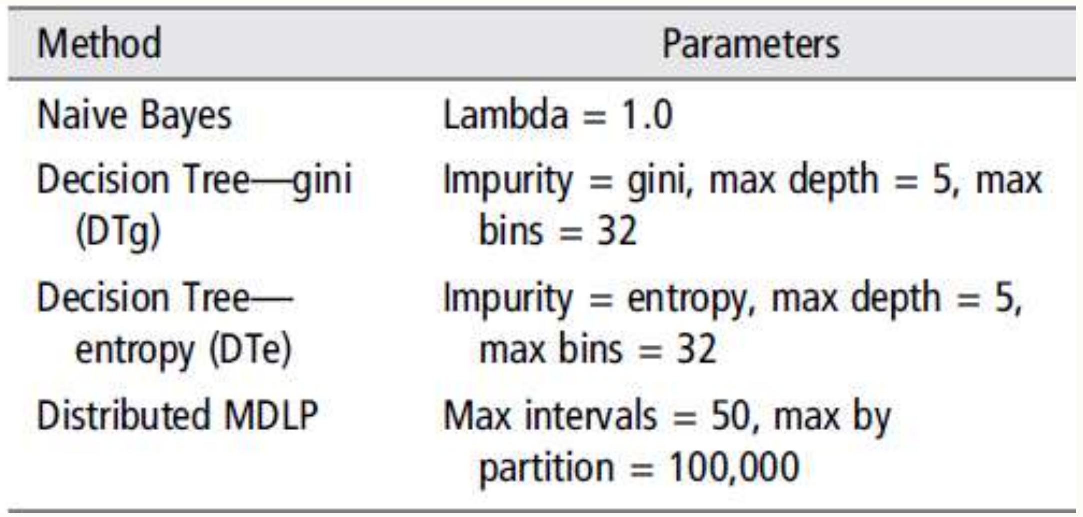
Map : in order to separate values by feature. It generates tuples with the value and the index for each feature as key and a class counter as value (< (A, A(s)), v >).

Reduce: the tuples are reduced using a function that aggregates all subsequent vectors with the same key, obtaining the class frequency for each distinct value in the dataset.

Experimental Results of DISTRIBUTED MDLP

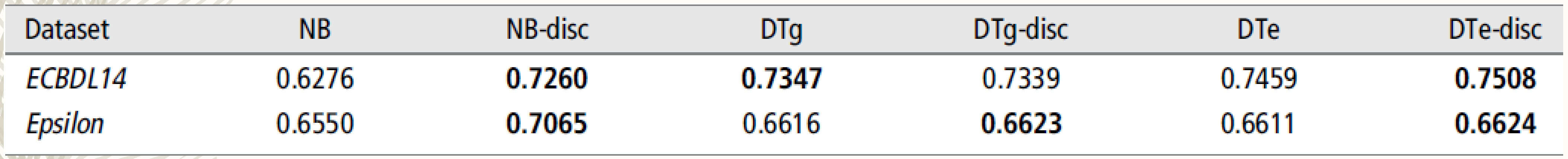
1. Discretization algorithm(DISTRIBUTED MDLP)as a preprocessing step leads to an improvement in classification accuracy with Naïve Bayes, for the two datasets tested.

2. For the other classifiers, our algorithm is capable of producing the same competitive results as those performed implicitly by the decision trees.



Method	Parameters
Naive Bayes	Lambda = 1.0
Decision Tree—gini (DTg)	Impurity = gini, max depth = 5, max bins = 32
Decision Tree—entropy (DTe)	Impurity = entropy, max depth = 5, max bins = 32
Distributed MDLP	Max intervals = 50, max by partition = 100,000

Classification Accuracy Values



Dataset	NB	NB-disc	DTg	DTg-disc	DTe	DTe-disc
ECBDL14	0.6276	**0.7260**	**0.7347**	0.7339	0.7459	**0.7508**
Epsilon	0.6550	**0.7065**	0.6616	**0.6623**	0.6611	**0.6624**

Total Experimental Results and Analysis

an analysis centered on the 30 discretizers studied is given as follows:

✓ Many classic discretizers are usually the best performing ones. This is the case of Chi -Merge, MDLP, Zeta, Distance, and Chi2.

✓ Other classic discretizers are not as good as they should be, considering that they have been improved over the years: Equal Width, Equal Frequency, 1R, ID3

✓ The empirical study allows us to stress several methods among the whole set:

- - FUSINTER, Chi Merge, CAIM, and Modified Chi2 offer excellent performances considering all types of classifiers.
- - PKID, FFD are suitable methods for lazy and Bayesian learning and CACC, Distance, and MODL are good choices in rule induction learning.
- - FUSINTER, Distance, Chi2, MDLP, and UCPD obtain a satisfactory tradeoff between the number of intervals produced and accuracy.

References

1)Salvador Garcı´a, Julia´n Luengo, Jose´ Antonio Sa´ ez, Victoria Lo´ pez, and Francisco Herrera
A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning

2)Sergio Ramírez-Gallego, Salvador García
Data discretization: taxonomy and big data challenge

3)Usama Fayyad,Keki Irani
Multi interval discretization of continuous attributes for classification learning