Automatic KDD Data Preparation Using Parallelism

Youssef Hmamouche*, Christian Ernst† and Alain Casali*
*LIF - CNRS UMR 6166, Aix Marseille Université, Marseille, France
Email: firstname.lastname@lif.univ-mrs.fr
†Email: chr29ernst@gmail.com

***Abstract*—We present an original framework for automatic data preparation, applicable in most Knowledge Discovery and Data Mining systems. It is based on the study of some statistical features of the target database samples. For each attribute of the database used, we automatically propose an optimized approach allowing to (i) detect and eliminate outliers, and (ii) to identify the most appropriate discretization method. Concerning the former, we show that the detection of an outlier depends on if data distribution is normal or not. When attempting to discern the appropriated discretization method, what is important is the shape followed by the density function of its distribution law. For this reason, we propose an automatic choice for finding the optimized discretization method, based on a multi-criteria (Entropy, Variance, Stability) evaluation. Most of the associated processings are performed in parallel, using the capabilities of multicore computers. Conducted experiments validate our approach, both on rule detection and on time series prediction. In particulary, we show that the same discretization method is not the best when applied to all the attributes of a specific database.**

***Keywords*–Data Mining; Data Preparation; Outliers detection and cleaning; Discretization Methods, Task parallelization.**

I. Introduction and Motivation

Data preparation in most of Knowledge and Discovery in Databases (KDD) systems has not been greatly developed in the literature. The single mining step is more often emphasized. And, when discussed, data preparation focuses most of the times on a single parameter (outlier detection and elimination, null values management, discretization method, *etc.*). Specific associated proposals only highlight on their advantages comparing themselves to others. There is no global nor automatic approach taking advantage of all of them. But the better data are prepared, the better results will be, and the faster mining algorithms will work.

In [1], we presented a global view of the whole data preparation process. Moreover, we proposed an automatization of most of the different steps of that process, based on the study of some statistical characteristics of the analysed database samples. This work was itself a continuation of the one exposed in [2]. In this latter, we proposed a simple but efficient approach to transform input data into a set of intervals (also called bins, clusters, classes, *etc.*). In a further step, we apply specific mining algorithms (correlation rules, *etc.*) on this set of bins. The very main difference with the former paper is that no automatization is performed. The parameters having an impact on data preparation have to be specified by the end-user before the data preparation process launches.

This paper in an extended version of [1]. Main improvements concern:

- A simplification and a better structuration of the presented concepts and processes;
- The use of parallelism in order to choose, when applicable, the most appropriate preparation method among different available methods;
- An expansion of our previous experiments. The ones concerning rule detection have been extended, and experimentations in order to forecast time series have been added.

The paper is organized as follows: Section II presents general aspects of data preparation. Section III and Section IV are dedicated to outlier detection and to discretization methods respectively. Each section is composed of two parts: (i) related work, and (ii) our approach. Section V discusses task parallelization possibilities. Here again, after introducing multicore programming, we present associated implementation issues concerning our work. In Section VI, we show the results of expanded experiments. Last section summarizes our contribution, and outlines some research perspectives.

II. Data Preparation

Raw input data must be prepared in any KDD system previous to the mining step. This is for two main reasons:

- If each value of each column is considered as a single item, there will be a combinatorial explosion of the search space, and thus very large response times;
- We cannot expect this task to be performed by hand because manual cleaning of data is time consuming and subject to many errors.

This step can be performed according to different method(ologie)s [3]. Nevertheless, it is generally divided into two tasks: (i) Preprocessing, and (ii) Transformation(s). When detailing hereafter these two tasks, focus is set on associated important parameters.

A. Preprocessing

Preprocessing consists in reducing the data structure by eliminating columns and rows of low significance [4].

a) Basic Column Elimination: Elimination of a column can be the result of, for example in the microelectronic industry, a sensor dysfunction, or the occurrence of a maintenance step; this implies that the sensor cannot transmit its values to the database. As a consequence, the associated column will contain many null/default values and must then be deleted from the input file. Elimination should be performed by using the Maximum Null Values ($MaxNV$) threshold. Furthermore, sometimes several sensors measure the same information, what produces identical columns in the database. In such a case, only a single column should be kept.

b) Elimination of Concentrated Data and Outliers: We first turn our attention to inconsistent values, such as "outliers" in noisy columns. Detection should be performed through another threshold (a convenient value of p when using the standardization method, see Section III-A). Found outliers are eliminated by forcing their values to Null. Another technique is to eliminate the columns that have a small standard deviation (threshold $MinStd$). Since their values are almost the same, we can assume that they do not have a significant impact on results; but their presence pollutes the search space and reduces response times. Similarly, the number of Distinct Values in each column should be bounded by the minimum ($MinDV$) and the maximum ($MaxDV$) values allowed.

B. Transformation

a) Data Normalization: This step is optional. It translates numeric values into a set of values comprised between 0 and 1. Standardizing data simplifies their classification.

b) Discretization: Discrete values deal with intervals of values, which are more concise to represent knowledge, so that they are easier to use and also more comprehensive than continuous values. Many discretization algorithms (*see* Section IV-A) have been proposed over the years for this. The number of used intervals ($NbBins$) as well as the selected discretization method among those available are here again parameters of the current step.

c) Pruning step: When the occurrence frequency of an interval is less than a given threshold ($MinSup$), then it is removed from the set of bins. If no bin remains in a column, then that column is entirely removed.

The presented thresholds/parameters are the ones we use for data preparation. In previous works, their values were fixed inside of a configuration file read by our software at setup. The main objective of this work is to automatically determine most of these variables without information loss. Focus is set in the two next sections on outlier and discretization management.

III. DETECTING OUTLIERS

An outlier is an atypical or erroneous value corresponding to a false measurement, an unwritten input, *etc.* Outlier detection is an uncontrolled problem because of values that deviate too greatly in comparison with the other data. In other words, they are associated with a significant deviation from the other observations [5]. In this section, we present some outlier detection methods associated to our approach using uni-variate data as input. We manage only uni-variate data because of the nature of our experimental data sets (*cf.* Section VI).

The following notations are used to describe outliers: X is a numeric attribute of a database relation, and is increasingly ordered. x is an arbitrary value, X_i is the i^{th} value, N is the number of values for X, σ its standard deviation, μ its mean, and s a central tendency parameter (variance, inter-quartile range, ...). X_1 and X_N are respectively the minimum and the maximum values of X. p is a probability, and k a parameter specified by the user, or computed by the system.

A. Related Work

We discuss hereafter four of the main uni-variate outlier detection methods.

Elimination after Standardizing the Distribution: This is the most conventional cleaning method [5]. It consists in taking into account σ and μ to determine the limits beyond which aberrant values are eliminated. For an arbitrary distribution, the inequality of Bienaymé-Tchebyshev indicates that the probability that the absolute deviation between a variable and its average is greater than k is less than or equal to $\frac{1}{k^2}$:

$$P(\left|\frac{x-\mu}{\sigma}\right| \geq k) \leq \frac{1}{k^2} \quad (1)$$

The idea is that we can set a threshold probability as a function of σ and μ above which we accept values as non-outliers. For example, with $k = 4.47$, the risk of considering that x, satisfying $\left|\frac{x-\mu}{\sigma}\right| \geq k$, is an outlier, is bounded by $\frac{1}{k^2} = 0.05$.

Algebraic Method: This method, presented in [6], uses the relative distance of a point to the "center" of the distribution, defined by: $d_i = \frac{|X_i-\mu|}{\sigma}$. Outliers are detected outside of the interval $[\mu - k \times Q_1, \mu + k \times Q_3]$, where k is generally fixed to 1.5, 2 or 3. Q_1 and Q_3 are the first and the third quartiles respectively.

Box Plot: This method, attributed to Tukey [7], is based on the difference between quartiles Q_1 and Q_3. It distinguishes two categories of extreme values determined outside the lower bound (LB) and the upper bound (UB):

$$\begin{cases} LB = Q_1 - k \times (Q_3 - Q_1) \\ UB = Q_3 + k \times (Q_3 - Q_1) \end{cases} \quad (2)$$

Grubbs' Test: Grubbs' method, presented in [8], is a statistical test for lower or higher abnormal data. It uses the difference between the average and the extreme values of the sample. The test is based on the assumption that the data have a normal distribution. The statistic used is: $T = max(\frac{X_N-\mu}{\sigma}, \frac{\mu-X_1}{\sigma})$. The assumption that the tested value (X_1 or X_N) is not an outlier is rejected at significance level α if:

$$T > \frac{N-1}{\sqrt{n}}\sqrt{\frac{\beta}{n-2\beta}} \quad (3)$$

where $\beta = t_{\alpha/(2n),n-2}$ is the quartile of order $\alpha/(2n)$ of the Student distribution with $n-2$ degrees of freedom.

B. An Original Method for Outlier Detection

Most of the existing outlier detection methods assume that the distribution is normal. However, in reality, many samples have asymmetric and multimodal distributions, and the use of these methods can have a significant influence at the data mining step. In such a case, each "distribution" has to be processed using an appropriated method. The considered approach consists in eliminating outliers in each column based on the normality of data, in order to minimize the risk of eliminating normal values.

Many tests have been proposed in the literature to evaluate the normality of a distribution: Kolmogorov-Smirnov [9], Shapiro-Wilks, Anderson-Darling, Jarque-Bera [10], *etc.* If the Kolmogorov-Smirnov test gives the best results whatever the distribution of the analysed data may be, it is nevertheless much more time consuming to compute then the others. This is why we have chosen the Jarque-Bera test (noted JB hereafter), much more simpler to implement as the others, as shown below:

$$JB = \frac{n}{6}({\gamma_3}^2 + \frac{\gamma_2^2}{4}) \quad (4)$$

This test follows a law of χ^2 with two degrees of freedom, and uses the *Skewness* $\gamma 3$ and the *Kurtosis* γ_2 statistics, defined respectively as follows:

$$\gamma_3 = E[(\frac{x-\mu}{\sigma})^3] \quad (5)$$

$$\gamma_2 = E[(\frac{x-\mu}{\sigma})^4] - 3 \quad (6)$$

If the JB normality test is not significant (the variable is normally distributed), then the Grubbs' test is used at a significance level of systematically 5%, otherwise the Box Plot method is used with parameter k automatically set to 3 in order to not to be too exhaustive toward outlier detection.

Figure 1 summarizes the process we chose to detect and eliminate outliers.

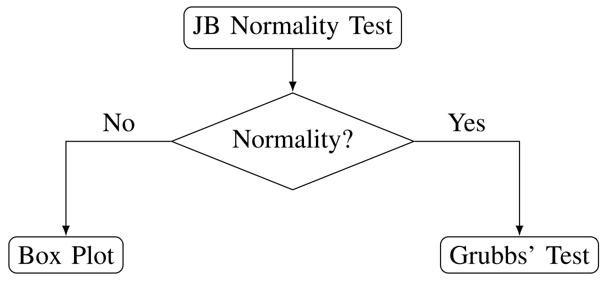


Figure 1: The outlier detection process.

Finally, the computation of $\gamma 3$ and $\gamma 2$ to evaluate the value of JB, so as other statistics needed by the Grubb's test and the Box Plot calculus, are performed in parallel in the manner shown in Listing 1 (*cf.* Section V). This in order to fasten the response times. Other statistics used in the next section are simultaneously collected here. Because the corresponding algorithm is very simple (the computation of each statistic is considered as a single task), we do not present it.

IV. Discretization Methods

Discretization of an attribute consists in finding $NbBins$ pairwise disjoint intervals that will further represent it in an efficient way. The final objective of discretization methods is to ensure that the mining part of the KDD process generates substantial results. In our approach, we only employ direct discretization methods in which $NbBins$ must be known in advance (and be the same for every column of the input data). $NbBins$ was in previous works a parameter fixed by the end-user. The literature proposes several formulas as an alternative (Rooks-Carruthers, Huntsberger, Scott, *etc.*) for computing such a number. Therefore, we switched to the Huntsberger formula, the most fitting from a theoretical point of view [11], and given by: $1 + 3.3 \times log_{10}(N)$.

A. Related Work

In this section, we only highlight the final discretization methods kept for this work. This is because the other tested methods have not revealed themselves to be as efficient as expected (such as Embedded Means Discretization), or are not a worthy alternative (such as Quantiles based Discretization) to the ones presented. In other words, the approach that we chose and which is discussed in the next sections, barely selected none of these alternative methods. Thus the methods we use are: Equal Width Discretization (EWD), Equal Frequency-Jenks Discretization (EFD-Jenks), AVerage and STandard deviation based discretization (AVST), and K-Means (KMEANS). These methods, which are unsupervised [12] and static [13], have been widely discussed in the literature: see for example [14] for EWD and AVST, [15] for EFD-Jenks, or [16] and [17] for KMEANS. For these reasons, we only summarize their main characteristics and their field of applicability in Table I.

TABLE I: SUMMARY OF THE DISCRETIZATION METHODS USED.

Method	Principle	Applicability
EWD	This simple to implement method creates intervals of equal width.	The approach cannot be applied to asymmetric or multi-modal distributions.
EFD-Jenks	Jenks' method provides classes with, if possible, the same number of values, while minimizing internal variance of intervals.	The method is effective from all statistical points of view but presents some complexity in the generation of the bins.
AVST	Bins are symmetrically centered around the mean and have a width equal to the standard deviation.	Intended only for normally distributed datasets.
KMEANS	Based on the Euclidean distance, this method determines a partition minimizing the quadratic error between the mean and the points of each interval.	Running time linear in $O(N \times NbBins \times k)$, where k in the number of iterations [?]. It is applicable to each form of distribution.

Let us underline that the upper limit fixed by the Huntsberger formula to the number of intervals to use is not always reached. It depends on the applied discretization method. Thus, EFD-Jenks and KMEANS methods generate most of the times less than $NbBins$ bins. This implies that other methods, which generate the $NbBins$ value differently for example through iteration steps, may apply if $NbBins$ can be upper bounded.

Example 1: Let us consider the numeric attribute $S_X = \{4.04, 5.13, 5.93, 6.81, 7.42, 9.26, 15.34, 17.89, 19.42, 24.40, 25.46, 26.37\}$. S_X contains 12 values, so by applying the Huntsberger's formula, if we aim to discretize this set, we have to use 4 bins.

Table II shows the bins obtained by applying all the discretization methods proposed in Table I. Figure 2 shows the number of values of S_X belonging to each bin associated to every discretization method.

As it is easy to understand, we cannot find two discretization methods producing the same set of bins. As a consequence, the distribution of the values of S_X is different depending on the method used.

B. Discretization Methods and Statistical Characteristics

When attempting to find the most appropriate discretization method for a column, what is important is not the law followed

TABLE II: SET OF BINS ASSOCIATED TO SAMPLE S_X.

Method	Bin$_1$	Bin$_2$	Bin$_3$	Bin$_4$
EWD	[4.04, 9.62[	[9.62, 15.21[	[15.21, 20.79[	[20.79, 26.37]
EFD-Jenks	[4.04; 5.94]	]5.94, 9.26]	]9.26, 19.42]	]19.42, 26.37]
AVST	[4.04; 5.53[	[5.53, 13.65[	[13.65, 21.78[	[21.78, 26.37]
KMEANS	[4.04; 6.37[	[6.37, 12.3[	[12.3, 22.95[	[22.95, 26.37]

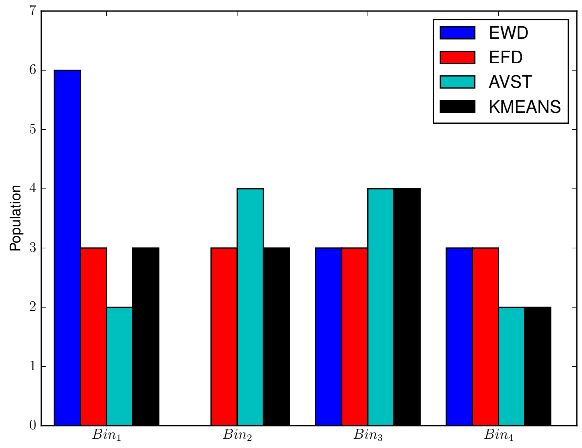


Figure 2: Population of each bin of sample S_X.

by its distribution, but the shape of its density function. This is why we first perform a descriptive analysis of the data in order to characterize, and finally to classify, each column according to normal, uniform, symmetric, antisymmetric or multimodal distributions. This is done in order to determine what discretization method(s) may apply. Concretely, we perform the following tests, which have to be carried out in the presented order:

1) We use the Kernel method introduced in [18] to characterize multimodal distributions. The method is based on estimating the density function of the sample by building a continuous function, and then calculating the number of peaks using its second derivative. This function allows us to approximate automatically the shape of the distribution. The multimodal distributions are those having a number of peaks strictly greater than 1.
2) To characterize antisymmetric and symmetric distributions in a next step, we use the skewness γ_3 (see formula (5)). The distribution is symmetric if $\gamma_3 = 0$. Practically, this rule is too exhaustive, so we relaxed it by imposing limits around 0 to set a fairly tolerant rule, which allows us to decide whether a distribution is considered antisymmetric or not. The associated method is based on a statistical test. The null hypothesis is that the distribution is symmetric. Consider the statistic: $T_{Skew} = \frac{N}{6}(\gamma_3^2)$. Under the null hypothesis, T_{Skew} follows a law of χ^2 with one degree of freedom. In this case, the distribution is antisymmetric with $\alpha = 5\%$ if $T_{Skew} > 3.8415$.
3) We use then the normalized Kurtosis, noted γ_2 (see formula (6)), to measure the peakedness of the distribution or the grouping of probability densities around the average, compared with the normal distribution. When γ_2 is close to zero, the distribution has a normalized peakedness.
A statistical test is used again to automatically decide whether the distribution has normalized peakedness or not. The null hypothesis is that the distribution has a normalized peakedness, and thus is uniform. Consider the statistic: $T_{Kurto} = \frac{N}{6}(\frac{\gamma_2^2}{4})$. Under the null hypothesis, T_{Kurto} follows a law of χ^2 with one degree of freedom. The null hypothesis is rejected at level of significance $\alpha = 0.05$ if $T_{Kurto} > 6.6349$.
4) To characterize normal distributions, we use the Jarque-Bera test (see equation (4) and relevant comments).

These four successive tests allow us to characterize the shape of the (density function of the) distribution of every column. Combined with the main characteristics of the discretization methods presented in the last section, we get Table III. This summarizes what discretization method(s) can be invoked depending on specific column statistics.

TABLE III: APPLICABILITY OF DISCRETIZATION METHODS DEPENDING ON THE DISTRIBUTION'S SHAPE.

	Normal	Uniform	Symm-etric	Antisym-metric	Multimodal
EWD	*	*	*		
EFD-Jenks	*	*	*	*	*
AVST	*				
KMEANS	*	*	*	*	*

Example 2: Continuing Example 1, the Kernel Density Estimation method [18] is used to build the density function of sample S_X (*cf.* Figure 3).

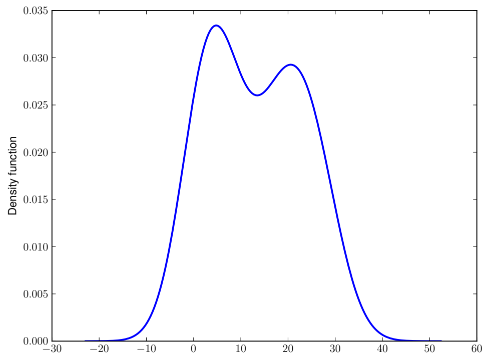


Figure 3: Density function of sample S_X using Kernel Density Estimation.

As we can see, the density function has two modes, is almost symmetric and normal. Since the density function is multimodal, we should stop at this point. But as shown in Table III, only EFD-Jenks and KMEANS produce interesting results according to our proposal. For the need of the example, let us perform the other tests. Since $\gamma_3 = -0.05$, the distribution is almost symmetric. As mentioned in (2), it depends on the threshold fixed if we consider that the distribution is symmetric or not. The distribution is not antisymmetric because $T_{Skew} = 0.005$. The distribution is not uniform since $\gamma_2 = -1.9$. As a consequence, $T_{Kurto} = 1.805$, and we have to reject the uniformity test. The Jarque-Berra test gives a p-value of 0.5191, which means that the sample is normal whatever the value set for α.

C. Multi-criteria Approach for Finding the Most Appropriate Discretization Method

Discretization must keep the initial statistical characteristics so as the homogeneity of the intervals, and reduce the size of the final data produced. Consequently, the discretization objectives are many and contradictory. For this reason, we chose a multi-criteria analysis to evaluate the available applicable methods of discretization. We use three criteria:

- The entropy H measures the uniformity of intervals. The higher the entropy, the more the discretization is adequate from the viewpoint of the number of elements in each interval:

$$H = -\sum_{i=1}^{NbBins} p_i \log_2(p_i) \tag{7}$$

  where p_i is the number of points of interval i divided by the total number of points (N), and $NbBins$ is the number of intervals. The maximum of H is computed by discretizing the attribute into $NbBins$ intervals with the same number of elements. In this case, H reduces to $log_2(NbBins)$.
- The index of variance J, introduced in [19], measures the interclass variances proportionally to the total variance. The closer the index is to 1, the more homogeneous the discretization is:

$$J = 1 - \frac{\text{Intra-intervals variance}}{\text{Total variance}}$$

- Finally, the stability S corresponds to the maximum distance between the distribution functions before and after discretization. Let F_1 and F_2 be the attribute distribution functions before and after discretization respectively:

$$S = sup_x(\left|F_1(x) - F_2(x)\right|) \tag{8}$$

The goal is to find solutions that present a compromise between the various performance measures. The evaluation of these methods should be done automatically, so we are in the category of *a priori* approaches, where the decision-maker intervenes just before the evaluation process step.

Aggregation methods are among the most widely used methods in multi-criteria analysis. The principle is to reduce to a unique criterion problem. In this category, the weighted sum method involves building a unique criterion function by associating a weight to each criterion [20], [21]. This method is limited by the choice of the weight, and requires comparable criteria. The method of inequality constraints is to maximize a single criterion by adding constraints to the values of the other criteria [22]. The disadvantage of this method is the choice of the thresholds of the added constraints.

Algorithm 1: MAD (Multi-criteria Analysis for Discretization)

```
Input: $X$ set of numeric values to discretize, DM set
       of discretization methods applicable
Output: best discretization method for $X$
1 foreach method $D \in$ DM do
2     Compute $V_D$;
3 end
4 return $argmin(V)$;
```

In our case, the alternatives are the 4 methods of discretization, and we discretize automatically columns separately, so the implementation facility is important in our approach. Hence the interest in using the aggregation method by reducing it to a unique criterion problem, by choosing the method that minimizes the Euclidean distance from the target point ($H = log_2(NbBins),\ J = 1,\ S = 0$).

Definition 1: Let D be an arbitrary discretization method. We can define V_D a measure of segmentation quality using the proposed multi-criteria analysis as follows:

$$V_D = \sqrt{(H_D - log_2(NbBins))^2 + (J_D - 1)^2 + S_D^2} \tag{9}$$

The following proposition is the main result of this article: It indicates how we chose the most appropriate discretization method among all the available ones.

Proposition 1: Let DM be a set of discretization methods; the set, noted $\mathbb{D}$, that minimizes V_{D} (*see* equation(9)), $\forall D \in \{DM\}$, contains the best discretization methods.

Corollary 1: The set of most appropriate discretization methods $\mathbb{D}$ can be obtained as follows:

$$\mathbb{D} = argmin(\{V_{\text{D}}, \forall D \in DM\}) \tag{10}$$

Let us underline that if $|\mathbb{D}| > 1$, then we have to choose one method among all. As a result of corollary 1, we propose the MAD (Multi-criteria Analysis for finding the best Discretization method) algorithm, see Algorithm 1.

Example 3: Continuing Example 1, Table IV shows the evaluation results for all the discretization methods at disposal. Let us underline that for the need of our example, all the values are computed for every discretization method, and not only for the ones that should have been selected after the step proposed in Section IV-B (*cf.* Table III).

TABLE IV: EVALUATION OF DISCRETIZATION METHODS.

	H	J	S	V_{DM}
EWD	1.5	0.972	0.25	0.559
EFD-Jenks	2	0.985	0.167	0.167
AVST	1.92	0.741	0.167	0.318
KMEANS	1.95	0.972	0.167	0.176

The results show that EFD-Jenks and KMEANS are the two methods that obtain the lowest values for V_D. The values got by the EWD and AVST methods are the worst: This is consistent with our optimization proposed in Table III, since the sample distribution is multimodal.

V. Parallelizing data preparation

Parallel architectures have become a standard today. As a result, applications can be distributed on several cores. Consequently, multicore applications run faster given that they require less process time to be executed, even if they may need on the other hand more memory for their data. But this latter inconvenient is minor when compared to the induced performances. We present in this section first some novel programming techniques, which allow to run easily different tasks in parallel. We show in a second step how we adapt these techniques to our work.

A. New Features in Multicore Encoding

Multicore processing is not a new concept, however only in the mid 2000s has the technology become mainstream with Intel and AMD. Moreover, since then, novel software environments that are able to take advantage simultaneously of the different existing processors have been designed (Cilk++, Open MP, TBB, *etc.*). They are based on the fact that looping functions are the key area where splitting parts of a loop across all available hardware resources increase application performance.

We focus hereafter on the relevant versions of the Microsoft .NET framework for C++ proposed since 2010. These enhance support for parallel programming by several utilities, among which the Task Parallel Library. This component entirely hides the multi-threading activity on the cores. The job of spawning and terminating threads, as well as scaling the number of threads according to the number of available cores, is done by the library itself.

The Parallel Patterns Library (PPL) is the corresponding available tool in the Visual C++ environment. The PPL operates on small units of work called Tasks. Each of them is defined by a λ calculus expression (see below). The PPL defines three kinds of facilities for parallel processing, where only templates for algorithms for parallel operations are of interest for this presentation.

Among the algorithms defined as templates for initiating parallel execution on multiple cores, we focus on the *parallel_invoke* algorithm used in the presented work (see end of Sections III-B and IV-C). It executes a set of two or more independent Tasks in parallel.

Another novelty introduced by the PPL is the use of λ expressions, now included in the C++11 language norm. These remove all need for scaffolding code, allowing a "function" to be defined in-line in another statement, as in the example provided by Listing 1. The λ element in the square brackets is called the capture specification. It relays to the compiler that a λ function is being created and that each local variable is being captured by reference (in our example). The final part is the function body.

```
// Returns the result of adding a value to itself
template <typename T> T twice(const T& t) {
        return t + t;
}
int n = 54; double d = 5.6; string s = "Hello";
// Call the function on each value concurrently
parallel_invoke(
        [&n] { n = twice(n); },
        [&d] { d = twice(d); },
        [&s] { s = twice(s); }
);
```

Listing 1: Parallel execution of 3 simple tasks

Listing 1 also shows the limits of parallelism. It is widely agreed that applications that may benefit from using more than one processor necessitate: (i) Operations that require a substantial amount of processor time, measured in seconds rather than milliseconds, and (ii), Operations that can be divided into significant units of calculation, which can be executed independently of one another. So the chosen example does not fit parallelization, but is used to illustrate the new features introduced by multicore programming techniques.

More details about parallel algorithms and the λ calculus can be found in [23], [24].

B. Application to data preparation

As a result of Table IV and of Proposition 1, we define the Pop (Parallel Optimized Preparation of data) method, see Algorithm 2.

For each attribute, after constructing Table III, each applicable discretization method is invoked and evaluated in order to keep finally the most appropriate. The content of these two tasks (three when involving the statistics computations) are executed in parallel using the *parallel_invoke* template (*cf.* previous section).
We discuss the advantages of this approach so as the got response times in the next section.

Algorithm 2: POP (Parallel Optimized Preparation of Data)

```
Input: $X$ set of numeric values to discretize, DM set
       of discretization methods applicable
Output: Best set of bins for $X$
1  Parallel_Invoke For each method $D \in$ DM do
2      Compute $\gamma_2$, $\gamma_3$ and perform Jarque-Bera test;
3  end
4  Parallel_Invoke For each method $D \in$ DM do
5      Remove $D$ from DM if it does not satisfy the
         criteria given in Table III;
6  end
7  Parallel_Invoke For each method $D \in$ DM do
8      Discretize $X$ according to $D$;
9      $V_D = \sqrt{(H_D - log_2(NbBins))^2 + (J_D - 1)^2 + S_D^2}$;
10 end
11 $\mathbb{D} = argmin(\{V_D, \forall D \in DM\})$;
12 return set of bins obtained in line 8 according to $\mathbb{D}$;
```

VI. Experimental Analysis

The goal of this section is to validate experimentally our approach according to two point of views: (i) firstly, we apply our methodology to the extraction of correlation and of association rules; (ii) secondly, we use it to forecast time series. These two application fields correspond to the two mainstream approaches in data mining, which consist in defining and using descriptive or predictive models. What means that the presented work can help to solve a great variety of associated problems.

A. Experimentation on rules detection

In this section, we present some experimental results by evaluating five samples. We decided to implement it using the MineCor KDD Software [2], but it could have been with another one (R Project, Tanagra, *etc.*). Sample$_1$ and Sample$_2$ correspond to real data, representing parameter (in the sense of attribute) measurements provided by microelectronics manufacturers after completion of the manufacturing process. The ultimate goal was here to detect correlations between one particular parameter (the yield) and the other attributes. Sample$_3$ is a randomly generated file that contains heterogeneous values. Sample$_4$ and Sample$_5$ are common data taken from the UCI Machine Learning Repository website [25]. Table V sums up the characteristics of the samples.

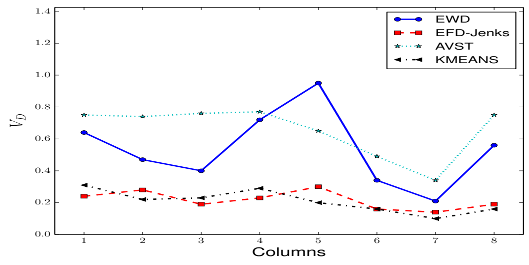
TABLE V: CHARACTERISTICS OF THE DATABASES USED.

Sample	**Number of columns**	**Number of rows**	**Type**
Sample$_1$ (amtel.csv)	8	727	real
Sample$_2$ (stm.csv)	1281	296	real
Sample$_3$ (generated.csv)	11	201	generated
Sample$_4$ (abalone.csv)	9	4177	real
Sample$_5$ (auto_mpg.csv)	8	398	real

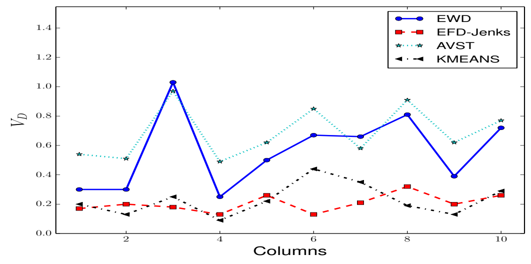
Experiments were performed on a 4 core computer (a DELL Workstation with a 2.8 GHz processor and 12 Gb RAM working under the Windows 7 64 bits OS). First, let us underline that we shall not focus in this section on performance issues. Of course, we have chosen to parallelize the underdone tasks in order to improve response times. As it is easy to understand, each of the *parallel_invoke* loops has a computational time closed to the most consuming calculus inside of each loop. Parallelism allows us to compute and then to evaluate different "possibilities" in order especially to chose the most efficient one for our purpose. This is done without waste of time, when comparing to a single "possibility" processing. Moreover, we can easily add other tasks to each parallelized loop (statistics computations, discretization methods, evaluation criteria). Some physical limits exist (currently): No more then seven tasks can be launched simultaneously within the 2010 C++ Microsoft .NET / PPL environment. But each individual described task does not require more than a few seconds to execute, even on the Sample$_2$ database.

Concerning outlier management, we recall that in the previous versions of our software (see [2]), we used the single standardization method with p set by the user (*cf.* Section III-A). With the new approach presented in Section III-B, we notice an improvement in the detection of true positive or false negative outliers by a factor of 2%.

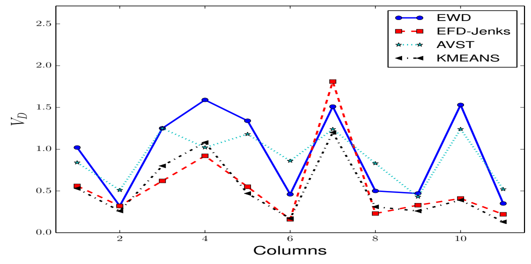
Figures 4 summarize the evaluation of the methods used on each of our samples, except on Sample$_2$: we have chosen to only show the results for the 10 first columns.



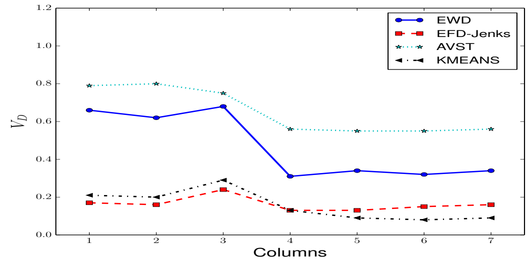
(a) Results for sample 1.



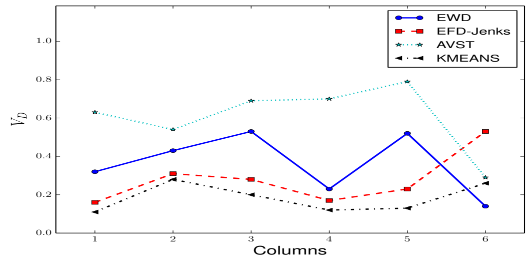
(b) Results for sample 2.



(c) Results for sample 3.



(d) Results for sample 4.



(e) Results for sample 5.

Figure 4: Discretization experimentations on the five samples.

For Sample$_1$ and Sample$_2$ attributes, which have symmetric and normal distributions, the evaluation on Figure 4a and 4b shows that the EFD-Jenks method provides generally the best results. The KMEANS method is unstable for these kinds of distributions, but sometimes provides the best discretization.

For the Sample$_3$ evaluation shown graphically in Figure 4c, the studied columns have relatively dispersed, asymmetric and multimodal distributions. “Best” discretizations are provided by EFD-Jenks and KMEANS methods. We note also that the EWD method is fast, and sometimes demonstrates good performances in comparison with the EFD-Jenks or KMEANS methods.

For Sample$_4$ and Sample$_5$ attributes, which distributions have a single mode and most of them are symmetric, the evaluation on Figures 4d and 4e shows that the KMEANS method provides generally the best results. The results given by EFD-Jenks method are closed to the ones obtained using KMEANS.

Finally, Figure 5 summarizes our approach. We have tested it over each column of each dataset. Any of the available methods is selected at least once in the dataset of the three first proposed samples (cf. Table V), which enforces our approach. As expected, EFD-Jenks is the method that is the most often kept by our software ($\simeq$ 42%). AVST and KMEANS are selected approximately a bit less than 30% each. EWD is only selected a very few times (less than 2%).

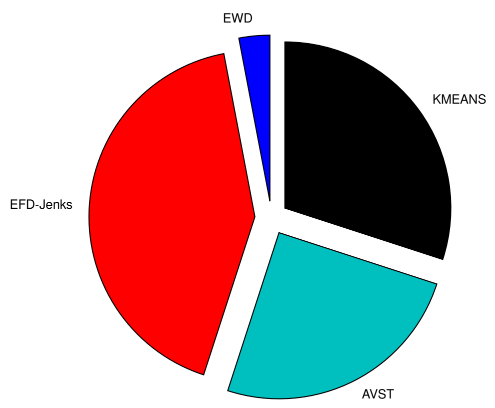
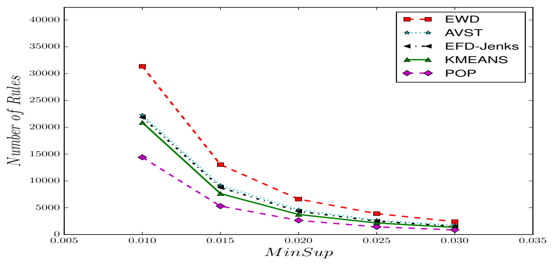
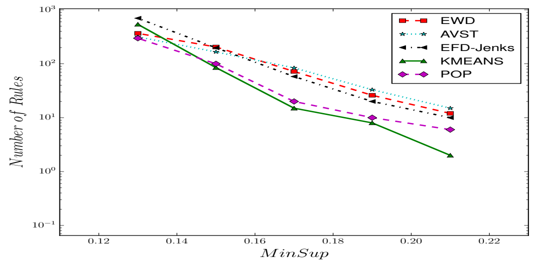


Figure 5: Global Distribution of DMs in our samples.

We focus hereafter on experiments performed in order to compare the different available discretization methods still on the three first samples. Figures 6a, 7a and 8a reference various experiments when mining Association Rules. Figures 6b, 7b and 8b correspond to experiments when mining Correlation Rules. When searching for Association Rules, the minimum confidence ($MinConf$) threshold has been arbitrarily set to 0.5. The different figures provide the number of Association or of Correlation Rules respectively, while the minimum support ($MinSup$) threshold varies. Each figure is composed of five curves. One for each of the four discretization methods presented in Table III, and one for our global method (POP). Each method is individually applied on each column of the considered database/dataset.

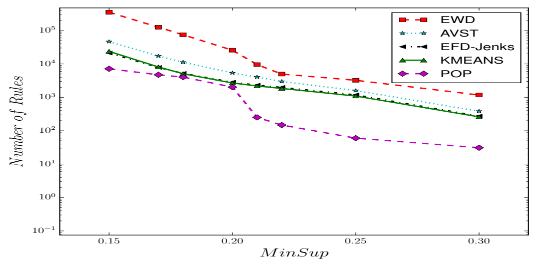


(a) Results for Apriori.

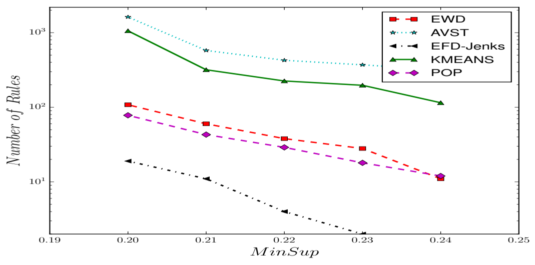


(b) Results for MineCor.

Figure 6: Execution on Sample$_1$.



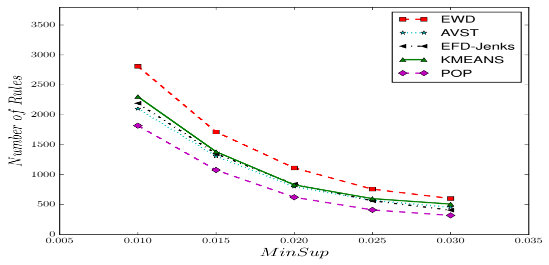
(a) Results for Apriori.



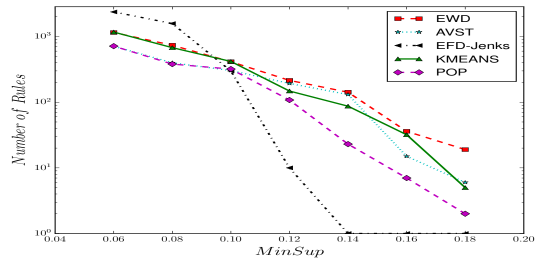
(b) Results for MineCor.

Figure 7: Execution on Sample$_2$.

Analyzing the Association Rules detection process, experiments show that POP gives the best results (few number of rules), and EWD is the worst. Using real data, the number of rules is reduced by a factor comprised between 5% and 20%. This reduction factor is even better using synthetic (generated) data and a low $MinSup$ threshold. When mining Correlation



(a) Results for Apriori.



(b) Results for MineCor.

Figure 8: Execution on Sample$_3$.

Rules on synthetic data, the method that gives the best results with high thresholds is KMEANS, while it is POP when the support is low. This can be explained by the fact that the generated data are sparse and multimodal. When examining the results on real databases, POP gives good results. However, let us underline that the EFD-Jenks method produces unexpected results: Either we have few rules (Figures 6a and 6b), or we have a lot (Figures 7a and 7b) with a low threshold. We suppose that the high number of used bins is at the basis of this result.

B. *Experimentation on time series forecasting*

In this section, we present an another practical application of the proposed method. It deals with the prediction of time series on financial data. Often, in time series prediction, interest is put on significant changes, instead of small fluctuations of the evolution of the data. Beside, in the machine learning field, the learning process for real data takes a substantial amount of time in the whole prediction process. For this reason, time series segmentation is used to make data more understandable by the prediction models, and to speed up the learning process. In light of that, the proposed method can be applied in order to help in the choice of the segmentation method.

For these experiments, we use a fixed prediction model (VAR-NN [26]), and multiple time series. We study hereafter the impact of the proposed methodology on the predictions.

1) The prediction model used: The prediction model used is first briefly described. The VAR-NN (Vector Auto-Regressive Neural Network) model, presented in [26], is a prediction model derived from the classical VAR (Vector Auto-Regressive) model [27], which is expressed as follows:

Let us consider a k-dimensional set of time series y_t, each one containing exactly T observations. The VAR(p) system expresses each variable of y_t as a linear function of the p previous values of itself and the p previous values of the other variables, plus an error term with a mean of zero.

$$y_t = \alpha_0 + \sum_{i=1}^{p} A_i y_{t-i} + \epsilon_t \tag{11}$$

ϵ_t is a white noise with a mean of zero, and $A_1, \ldots, A_p$ are $(k \times k)$ matrices parameters of the model. The general expression of the non linear VAR model is different from the classical model in the way that the parameters of the model values are not linear.

$$y_t = F_t(y_{t-1}, y_{t-2}, ..., y_{t-p} + x_{t-1}, x_{t-2}, ..., x_{t-p}) \tag{12}$$

We use in this experiment the VAR-NN (Vector Auto-Regressive Neural Network) model [28], with multi-layer perceptron structure, and based on the back-propagation algorithm. An example of two time series as an input of the network, with one hidden layer, is given in Figure 9.

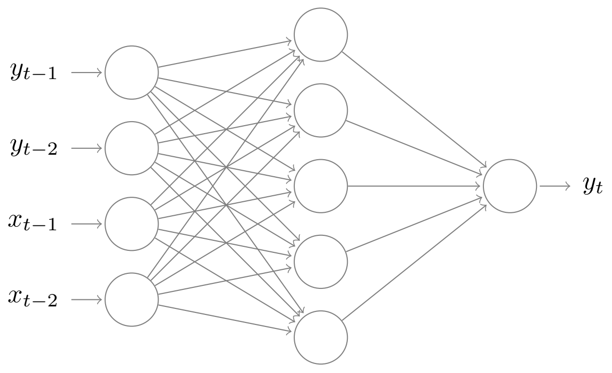


Figure 9: Illustration of a bivariate VAR-NN model with a lag parameter $p = 2$ and with one hidden layer.

2) Time series used: We use the following financial time series:

- ts_1: Financial french time series expressing the prices of 9 articles containing (Oil, Propane, Gold, euros/dollars, Butane, Cac40) and others, from what prices have been extracted between 2013/03/12 and 2016/03/01.
- ts_2 (w.tb3n6ms): weekly 3 and 6 months US Treasury Bill interest rates from 1958/12/12 until 2004/08/06, extracted from the R package **FinTS** [29].
- ts_3 (m.fac.9003): object of 168 observations giving simple excess returns of 13 stocks and the Standard and Poors 500 index over the monthly series of three-months Treasury Bill rates of the secondary market as the risk-free rate from January 1990 to December 2003, extracted from the **R** package **FinTS**.

3) Experimentations: Let p be the lag parameter of the VAR-NN model, setted in our case according to the length of the series (see Table VI), and $NbVar$ the number of the variables of the multivariate time series to predict. We use a neural network with (i) 10000 as maximum of iterations,

TABLE VI: CHARACTERISTICS OF THE TIME SERIES USED.

Time series	**Number of attributes**	**Number of rows**	**Number of predictions**	**Lag parameter**	**Type**
ts_1	9	1090	100	20	real
ts_2	2	2383	200	40	real
ts_3	14	168	20	9	real

TABLE VII: Detection of the best methods for both, the discretization quality and the prediction precision

Time series	**Attributs**	**Forecasting score** $(1000.MSE)$				**Discretization evaluation** (V_d)				**best discretization**	**best forecaster**
		ewd	**efd**	**avst**	**kmeans**	**ewd**	**efd**	**avst**	**kmeans**		
ts_1	**col1**	0.9	**0.4**	10.5	0.6	0.42	**0.22**	0.52	1.07	efd	efd
	col2	0.6	**0.1**	3.1	0.2	0.60	**0.18**	0.86	0.53	efd	efd
	col3	4	**2.9**	5.7	3	0.28	**0.10**	0.40	0.26	efd	efd
	col4	4.1	**3**	5.8	3.4	0.29	**0.10**	0.40	0.25	efd	efd
	col5	2.1	1.1	1.8	**0.9**	0.63	0.29	0.56	**0.28**	kmeans	kmeans
	col6	2.2	**1.1**	6.5	1.8	0.52	**0.21**	0.58	0.25	efd	efd
	col7	1.1	**0.3**	2.6	0.6	0.36	**0.18**	0.47	0.53	efd	efd
	col8	0.9	**0.5**	3	**0.5**	0.65	**0.24**	000.61	0.59	efd,kmeans	efd
	col9	3.2	**2.4**	11.6	**2.6**	0.23	0.12	000.62	**0.11**	efd	kmeans
ts_2	**col1**	**1.5**	1.7	6.5	2.8	0.48	**0.16**	0.61	0.30	efd	ewd
	col2	**1.4**	1.5	6.1	1.8	0.47	0.29	0.62	**0.22**	kmeans	ewd
ts_3	**col1**	104.5	108.4	89.9	**67**	0.53	0.23	0.52	**000.13**	kmeans	kmeans
	col2	57.7	65.5	75.1	**44.8**	0.61	0.35	0.76	**000.22**	kmeans	kmeans
	col3	93.4	**67.7**	83	76.5	0.46	**0.13**	0.57	000.16	efd	efd
	col4	127.7	120.6	131	**91.1**	0.28	0.20	0.51	**000.10**	kmeans	kmeans
	col5	91.7	**80.6**	78.9	96.6	0.33	0.28	0.52	**000.18**	kmeans	efd
	col6	113.6	122.7	**85.8**	97.2	0.48	0.26	0.58	**000.17**	kmeans	avst
	col7	105.8	**92.3**	102.6	110.2	0.53	0.22	0.56	**000.11**	kmeans	efd
	col8	84.6	**64.1**	73.8	79.5	0.58	**0.23**	0.60	**000.22**	kmeans,efd	efd
	col9	**66.9**	78.8	90.5	92.4	0.65	0.38	0.92	**000.33**	kmeans	efd
	col10	**41.3**	56	55.6	48.6	0.61	0.28	0.74	**000.15**	kmeans	ewd
	col11	**35.9**	47.6	39.1	36.1	0.69	**0.27**	0.81	000.34	efd	ewd
	col12	72.3	50.4	59.6	**42.5**	0.65	0.29	0.74	**000.21**	kmeans	kmeans
	col13	**98.7**	120	102.8	107	0.43	**0.16**	0.50	**000.15**	kmeans	ewd
	col14	**58.5**	68.4	94	105.8	0.56	0.25	0.76	**000.14**	kmeans	ewd

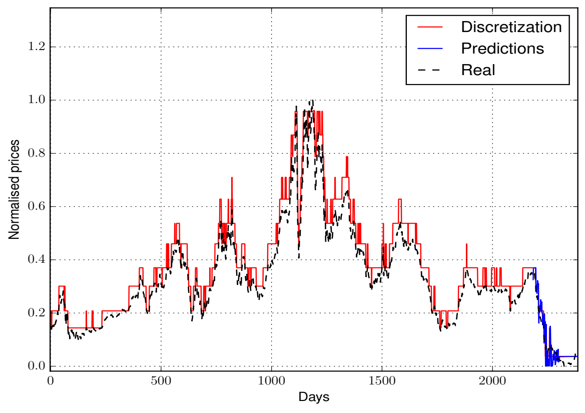
(ii) 4 hidden layers of size $(2/3, 1/4, 1/4, 1/4) \times k$, where $k = p \times NbVar$, is the number of inputs of the model (since we use the p previous values of $NbVar$ variables).

First, we apply the discretization methods (EWD, EFD-Jenks, AVST, KMEANS) on the time series, in order to find the best one according to formula (9). Then we select the best method for each attribute in terms of the predictions precision. And finally, we compare the results for both discretization and prediction. The learning step of the prediction model is performed on the time series without a fixed number of last values (for which we make predictions). These are setted depending on the length of the series as shown in Table VI. Experiments are made in forecasting the last values as a sliding window. Each time we make a prediction, we learn from the real one, and so on. Finally, after obtaining all the predictions, we calculate the MSE (Mean Squared Error) of the predictions. The results of finding the best methods for both, the discretization quality using the proposed multicriteria approach, and the precision of the prediction, are summarized in Table VII. We show in Figure 10 the real, discretization and predictions of one target variable among 25 possibilities. The results of the evaluations of all the attributes are summarized in Table VII.

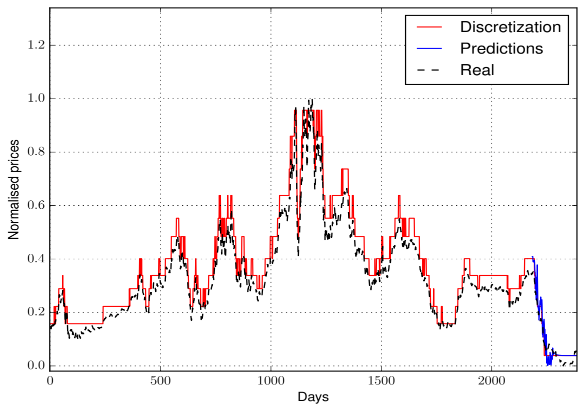
4) Interpretation: The evaluations illustrated in Table VII show that there is a rightness of 56% between best methods of discretization and predictions. Even if the best method of discretization is not always the best predictor, it shows a good score of prediction compared with the best one. What means that the multi-criteria evaluation of the discretization methods can predict at 56% the methods that will give the best predictions, and this just basing on the statistical characteristics of the discretized series. Consequently, we demonstrate that there is an impact justified by the evaluation made on financial time series with different lengths and different variables.

VII. Conclusion and future work

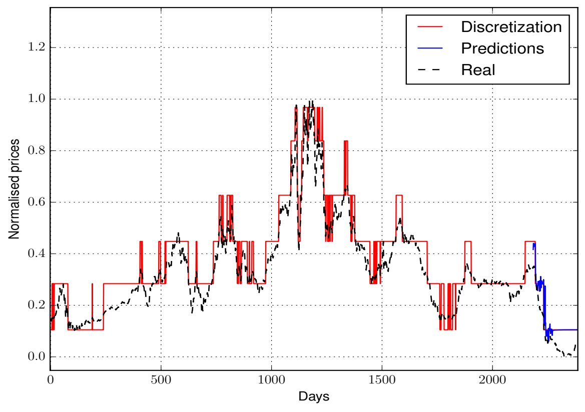
In this paper, we presented a new approach for automatic data preparation implementable in most of KDD systems. This step is generally split into two sub-steps: (i) detecting and eliminating the outliers, and (ii) applying a discretization method in order to transform any column into a set of clusters. In this article, we show that the detection of outliers depends on the knowledge of the data distribution (normal or not). As a consequence, we do not have to apply the same pruning method (Box plot *vs.* Grubb's test). Moreover, when trying to find the most appropriate discretization method, what is important is not the law followed by the column, but the shape of its density function. This is why we propose an automatic choice for finding the most appropriate discretization method based on a multi-criteria approach, according to several criteria (Entropy, Variance, Stability). Experiments tasks are performed using multicore programming. What allows us to explore different solutions, to evaluate them, and to keep the most appropriated one for the studied data set without waste of time. As main result, experimental evaluations done both on real and synthetic data, and for different mining objectives,



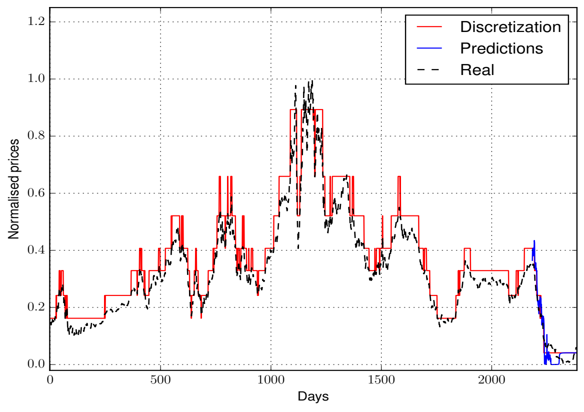
(a) Results of EWD method.



(b) Results of EFD-Jenks method.



(c) Results of AVST method.



(d) Results of KMEANS method.

Figure 10: Predictions, discretized and real values for the target attribut Col2/ts_2.

validate our work, showing that it is not always the very same discretization method that is the best: Each method has its strengths and drawbacks. Moreover, experiments performed, on one hand when mining correlation rules, show a significant reduction of the number of produced rules, and, on the other hand when forecasting times series, show a significant improvement of the predictions obtained. We can conclude that our methodology produces better result in most cases.

For future works, we aim to experimentally validate the relationship between the distribution shape and the applicability of used methods, to add other discretization methods (Khiops, Chimerge, Entropy Minimization Discretization, *etc.*) to our system, and to understand why our methodology does not give always the best result in order to improve it.

References

[1] Y. Hmamouche, C. Ernst, and A. Casali, "Automatic kdd data preparation using multi-criteria features," in IMMM 2015, The Fifth International Conference on Advances in Information Mining and Management, 2015, pp. 33–38.

[2] C. Ernst and A. Casali, "Data preparation in the minecor kdd framework," in IMMM 2011, The First International Conference on Advances in Information Mining and Management, 2011, pp. 16–22.

[3] D. Pyle, Data Preparation for Data Mining. Morgan Kaufmann, 1999.

[4] O. Stepankova, P. Aubrecht, Z. Kouba, and P. Miksovsky, "Preprocessing for data mining and decision support," in Data Mining and Decision Support: Integration and Collaboration, K. A. Publishers, Ed., 2003, pp. 107–117.

[5] C. Aggarwal and P. Yu, "Outlier detection for high dimensional data," in SIGMOD Conference, S. Mehrotra and T. K. Sellis, Eds. ACM, 2001, pp. 37–46.

[6] M. Grun-Rehomme, O. Vasechko et al., "Méthodes de détection des unités atypiques: Cas des enquêtes structurelles ukrainiennes," in 42èmes Journées de Statistique, 2010.

[7] J. W. Tukey, "Exploratory data analysis. 1977," Massachusetts: Addison-Wesley, 1976.

[8] F. E. Grubbs, "Procedures for detecting outlying observations in samples," Technometrics, vol. 11, no. 1, 1969, pp. 1–21.

[9] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," Journal of the American Statistical Association, vol. 62, no. 318, 1967, pp. 399–402.

[10] C. M. Jarque and A. K. Bera, "Efficient tests for normality, homoscedas-

ticity and serial independence of regression residuals," Economics Letters, vol. 6, no. 3, 1980, pp. 255–259.

[11] C. Cauvin, F. Escobar, and A. Serradj, Thematic Cartography, Cartography and the Impact of the Quantitative Revolution. John Wiley & Sons, 2013, vol. 2.

[12] I. Kononenko and S. J. Hong, "Attribute selection for modelling," Future Generation Computer Systems, vol. 13, no. 2, 1997, pp. 181–195.

[13] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," GESTS International Transactions on Computer Science and Engineering, vol. 32, no. 1, 2006, pp. 47–58.

[14] J. W. Grzymala-Busse, "Discretization based on entropy and multiple scanning," Entropy, vol. 15, no. 5, 2013, pp. 1486–1502.

[15] G. Jenks, "The data model concept in statistical mapping," in International Yearbook of Cartography, vol. 7, 1967, pp. 186–190.

[16] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 7, 2002, pp. 881–892.

[17] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recognition Letters, vol. 31, no. 8, 2010, pp. 651–666.

[18] B. W. Silverman, Density estimation for statistics and data analysis. CRC press, 1986, vol. 26.

[19] C. Cauvin, F. Escobar, and A. Serradj, Cartographie thématique. 3. Méthodes quantitatives et transformations attributaires. Lavoisier, 2008.

[20] P. M. Pardalos, Y. Siskos, and C. Zopounidis, Advances in multicriteria analysis. Springer, 1995.

[21] B. Roy and P. Vincke, "Multicriteria analysis: survey and new directions," European Journal of Operational Research, vol. 8, no. 3, 1981, pp. 207–218.

[22] C. Zopounidis and P. M. Pardalos, Handbook of multicriteria analysis. Springer Science & Business Media, 2010, vol. 103.

[23] A. Casali and C. Ernst, "Extracting correlated patterns on multicore architectures," in Availability, Reliability, and Security in Information Systems and HCI - IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES 2013, Regensburg, Germany, September 2-6, 2013. Proceedings, 2013, pp. 118–133.

[24] C. Ernst, Y. Hmamouche, and A. Casali, "Pop: A parallel optimized preparation of data for data mining," in 2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), vol. 1. IEEE, 2015, pp. 36–45.

[25] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[26] D. U. Wutsqa, S. G. Subanar, and Z. Sujuti, "Forecasting performance of var-nn and varma models," in Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, 2006.

[27] D. C. Montgomery, C. L. Jennings, and M. Kulahci, Introduction to time series analysis and forecasting. John Wiley & Sons, 2015.

[28] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, 2015, pp. 85–117.

[29] S. Graves, "The fints package," 2008. [Online]. Available: https://cran.r-project.org/web/packages/FinTS/