Data Preprocessing for Supervised Leaning

S. B. Kotsiantis, D. Kanellopoulos and P. E. Pintelas

***Abstract*—** Many factors affect the success of Machine Learning (ML) on a given task. The representation and quality of the instance data is first and foremost. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. It is well known that data preparation and filtering steps take considerable amount of processing time in ML problems. Data pre-processing includes data cleaning, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set. It would be nice if a single sequence of data pre-processing algorithms had the best performance for each data set but this is not happened. Thus, we present the most well know algorithms for each step of data pre-processing so that one achieves the best performance for their data set.

***Keywords*—**data mining, feature selection, data cleaning

I. Introduction

THE data preprocessing can often have a significant impact on generalization performance of a supervised ML algorithm. The elimination of noise instances is one of the most difficult problems in inductive ML [48]. Usually the removed instances have excessively deviating instances that have too many null feature values. These excessively deviating features are also referred to as outliers. In addition, a common approach to cope with the infeasibility of learning from very large data sets is to select a single sample from the large data set. Missing data handling is another issue often dealt with in the data preparation steps.

The symbolic, logical learning algorithms are able to process symbolic, categorical data only. However, real-world problems involve both symbolic and numerical features. Therefore, there is an important issue to discretize numerical (continuous) features. Grouping of values of symbolic features is a useful process, too [18]. It is a known problem that features with too many values are overestimated in the process of selecting the most informative features, both for inducing decision trees and for deriving decision rules.

Moreover, in real-world data, the representation of data often uses too many features, but only a few of them may be related to the target concept. There may be redundancy, where certain features are correlated so that is not necessary to include all of them in modeling; and interdependence, where two or more features between them convey important information that is obscure if any of them is included on its own [15]. Feature subset selection is the process of identifying and removing as much irrelevant and redundant information as possible. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept. Furthermore, the problem of feature interaction can be addressed by constructing new features from the basic feature set. Transformed features generated by feature construction may provide a better discriminative ability than the best subset of given features

This paper addresses issues of data pre-processing that can have a significant impact on generalization performance of a ML algorithm. We present the most well know algorithms for each step of data pre-processing so that one achieves the best performance for their data set.

The next section covers instance selection and outliers detection. The topic of processing unknown feature values is described in section 3. The problem of choosing the interval borders and the correct arity (the number of categorical values) for the discretization is covered in section 4. The section 5 explains the data normalization techniques (such as scaling down transformation of the features) that are important for many neural network and k-Nearest Neighbourhood algorithms, while the section 6 describes the Feature Selection (FS) methods. Finally, the feature construction algorithms are covered in section 7 and the closing section concludes this work.

II. Instance Selection and Outliers Detection

Generally, instance selection approaches are distinguished between *filter* and *wrapper* [13], [21]. Filter evaluation only considers data reduction but does not take into account activities. On contrary, wrapper approaches explicitly emphasize the ML aspect and evaluate results by using the specific ML algorithm to trigger instance selection.

Variable-by-variable data cleaning is straightforward filter approach (those values that are suspicious due to their relationship to a specific probability distribution, say a normal distribution with a mean of 5, a standard deviation of 3, and a suspicious value of 10). Table 1 shows examples of how this metadata can help on detecting a number of possible data quality problems. Moreover, a number of authors focused on

Manuscript received Feb 19, 2006. The Project is Co-Funded by the European Social Fund & National Resources - EPEAEK II.

S. B. Kotsiantis is with Educational Software Development Laboratory, University of Patras, Greece (phone: +302610997833; fax: +302610997313; e-mail: sotos@ math.upatras.gr).

D. Kanellopoulos is with Educational Software Development Laboratory, University of Patras, Greece (e-mail: dkanellop@ teipat.gr).

P. E. Pintelas is with Educational Software Development Laboratory, University of Patras, Greece (e-mail: pintelas@ math.upatras.gr).

the problem of duplicate instance identification and elimination, e.g., [16].

TABLE I
EXAMPLES FOR THE USE OF VARIABLE-BY-VARIABLE DATA CLEANING

Problems	*Metadata*	*Examples/Heuristics*
Illegal values	cardinality	e.g., cardinality (gender)> 2 indicates problem
	max, min	max, min should not be outside of permissible range
	variance, deviation	variance, deviation of statistical values should not be higher than threshold
Misspellings	feature values	sorting on values often brings misspelled values next to correct values

An *inlier* is a data value that lies in the interior of a statistical distribution and is in error. Because inliers are difficult to distinguish from good data values they are sometimes difficult to find and correct. Multivariate data cleaning is more difficult, but is an essential step in a complete analysis [43]. Examples are the distance based outlier detection algorithm RT [22] and the density based outliers LOF [3].

Brodley and Friedl [4] focus on wrapper approach with improving the quality of training data by identifying and eliminating mislabelled instances prior to applying the chosen ML algorithm. Their first step is to identify candidate instances by using *m* learning algorithms to tag instances as correctly or incorrectly labelled. The second step is to form a classifier using a new version of the training data for which all of the instances identified as mislabelled are removed. Filtering can be based on one or more of the *m* base level classifiers' tags.

However, instance selection isn't only used to handle noise but for coping with the infeasibility of learning from very large data sets. Instance selection in this case is an optimization problem that attempts to maintain the mining quality while minimizing the sample size [33]. It reduces data and enables a learning algorithm to function and work effectively with huge data. There is a variety of procedures for sampling instances from a large data set. The most well known are [6]:

- *Random sampling* that selects a subset of instances randomly.
- *Stratified sampling* that is applicable when the class values are not uniformly distributed in the training sets. Instances of the minority class(es) are selected with a greater frequency in order to even out the distribution.

Sampling is well accepted by the statistics community, who observe that "a powerful computationally intense procedure operating on a sub-sample of the data may in fact provide superior accuracy than a less sophisticated one using the entire data base" [12]. In practice, as the amount of data grows, the rate of increase in accuracy slows, forming the familiar learning curve. Whether sampling will be effective depends on how dramatically the rate of increase slows. Oates and Jensen [37] studied decision tree induction for nineteen data sets, and looked specifically at the number of instances necessary before the learning curves reached a plateau. Surprisingly, for these nineteen data sets, a plateau was reached after very few training instances.

Reinartz [42] presents a unifying framework, which covers individual state of the art approaches related to instance selection. First, an application of a statistical sampling technique draws an initial sample. In the next step, a clustering technique groups the initial sample into subsets of similar instances. For each of these subsets, the prototyping step selects or constructs a smaller set of representative prototypes. The set of prototypes then constitutes the final output of instance selection.

Typically learners are expected to be able to generalize over unseen instances of any class with equal accuracy. That is, in a two class domain of positive and negative examples, the learner will perform on an unseen set of examples with equal accuracy on both the positive and negative classes. This of course is the ideal situation. In many applications learners are faced with imbalanced data sets, which can cause the learner to be biased towards one class. This bias is the result of one class being heavily under represented in the training data compared to the other classes. It can be attributed to two factors that relate to the way in which learners are designed: Inductive learners are typically designed to minimize errors over the training examples. Classes containing few examples can be largely ignored by learning algorithms because the cost of performing well on the over-represented class outweighs the cost of doing poorly on the smaller class. Another factor contributing to the bias is over-fitting. Over-fitting occurs when a learning algorithm creates a hypothesis that performs well over the training data but does not generalize well over unseen data. This can occur on an under represented class because the learning algorithm creates a hypothesis that can easily fit a small number of examples, but it fits them too specifically.

Imbalanced data sets have recently received attention in the machine learning community. Common solutions of instance selection include:

- Duplicating training examples of the under represented class [30]. This is in effect re-sampling the examples and will be referred to in this paper as over-sampling.
- Removing training examples of the over represented class [26]. This is referred to as downsizing to reflect that the overall size of the data set is smaller after this balancing technique has taken place.

III. MISSING FEATURE VALUES

Incomplete data is an unavoidable problem in dealing with most of the real world data sources. The topic has been discussed and analyzed by several researchers in the field of ML [5], [14]. Generally, there are some important factors to be taken into account when processing unknown feature values. One of the most important ones is the source of 'unknownness': (i) a value is missing because it was forgotten or lost; (ii) a certain feature is not applicable for a given instance, e.g., it does not exist for a given instance; (iii) for a given observation, the designer of a training set does not care

about the value of a certain feature (so-called don't-care value).

Analogically with the case, the expert has to choose from a number of methods for handling missing data [27]:

- *Method of Ignoring Instances with Unknown Feature Values:* This method is the simplest: just ignore the instances, which have at least one unknown feature value.
- *Most Common Feature Value:* The value of the feature that occurs most often is selected to be the value for all the unknown values of the feature.
- *Concept Most Common Feature Value:* This time the value of the feature, which occurs the most common within the same class is selected to be the value for all the unknown values of the feature.
- *Mean substitution:* Substitute a feature's mean value computed from available cases to fill in missing data values on the remaining cases. A smarter solution than using the "general" feature mean is to use the feature mean for all samples belonging to the same class to fill in the missing value
- *Regression or classification methods:* Develop a regression or classification model based on complete case data for a given feature, treating it as the outcome and using all other relevant features as predictors.
- *Hot deck imputation:* Identify the most similar case to the case with a missing value and substitute the most similar case's Y value for the missing case's Y value.
- *Method of Treating Missing Feature Values as Special Values:* treating "unknown" itself as a new value for the features that contain missing values.

IV. Discretization

Discretization should significantly reduce the number of possible values of the continuous feature since large number of possible feature values contributes to slow and ineffective process of inductive ML. The problem of choosing the interval borders and the correct arity for the discretization of a numerical value range remains an open problem in numerical feature handling. The typical discretization process is presented in the Fig. 1.

Generally, discretization algorithms can be divided into unsupervised algorithms that discretize attributes without taking into account the class labels and supervised algorithms that discretize attributes by taking into account the class-attribute [34]. The simplest discretization method is an unsupervised direct method named equal size discretization. It calculates the maximum and the minimum for the feature that is being discretized and partitions the range observed into k equal sized intervals. Equal frequency is another unsupervised method. It counts the number of values we have from the feature that we are trying to discretize and partitions it into intervals containing the same number of instances.

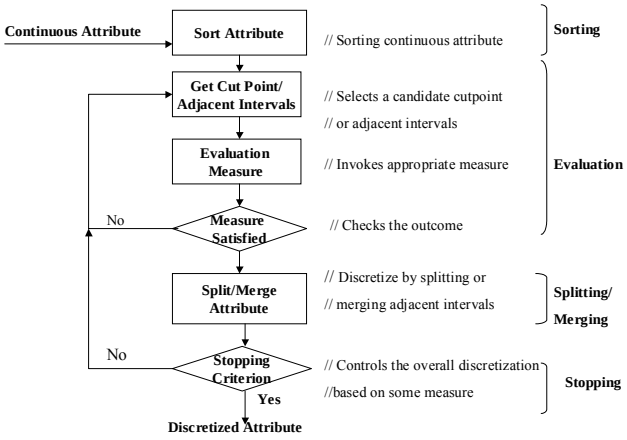

Fig. 1 Discretization process

Most discretization methods are divided into top-down and bottom-up methods. Top down methods start from the initial interval and recursively split it into smaller intervals. Bottom-up methods start from the set of single value intervals and iteratively merge neighboring intervals. Some of these methods require user parameters to modify the behavior of the discretization criterion or to set up a threshold for the stopping rule. Boulle [2] presented a recent discretization method named Khiops. This is a bottom-up method based on the global optimization of chi-square.

Moreover, error-based methods, for example Maas [35], evaluate candidate cut points against an error function and explore a search space of boundary points to minimize the sum of false positive and false negative errors on the training set. Entropy is another supervised incremental top down method described in [11]. Entropy discretization recursively selects the cut-points minimizing entropy until a stopping criterion based on the Minimum Description Length criterion ends the recursion.

Static methods, such as binning and entropy-based partitioning, determine the number of partitions for each feature independent of the other features. On the other hand, dynamic methods [38] conduct a search through the space of possible *k* partitions for all features simultaneously, thereby capturing interdependencies in feature discretization. Kohavi and Sahami [23] have compared static discretization with dynamic methods using cross-validation to estimate the accuracy of different values of *k*. However, they report no significant improvement in employing dynamic discretization over static methods.

V. Data Normalization

Normalization is a "scaling down" transformation of the features. Within a feature there is often a large difference between the maximum and minimum values, e.g. 0.01 and 1000. When normalization is performed the value magnitudes and scaled to appreciably low values. This is important for many neural network and k-Nearest Neighbourhood algorithms. The two most common methods for this scope are:

- min-max normalization:
  $v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$
- z-score normalization: $v' = \frac{v - mean_A}{stand\_dev_A}$

where v is the old feature value and v' the new one.

VI. Feature Selection

Feature subset selection is the process of identifying and removing as much irrelevant and redundant features as possible (see Fig. 2). This reduces the dimensionality of the data and enables learning algorithms to operate faster and more effectively. Generally, features are characterized as:

- Relevant: These are features have an influence on the output and their role can not be assumed by the rest
- Irrelevant: Irrelevant features are defined as those features not having any influence on the output, and whose values are generated at random for each example.
- Redundant: A redundancy exists whenever a feature can take the role of another (perhaps the simplest way to model redundancy).

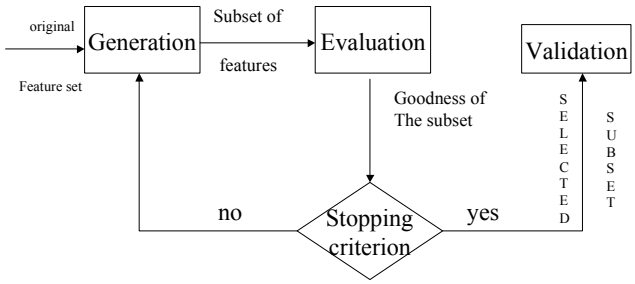

Fig. 2 Feature subset selection

FS algorithms in general have two components [20]: a selection algorithm that generates proposed subsets of features and attempts to find an optimal subset; and an evaluation algorithm that determines how 'good' a proposed feature subset is, returning some measure of goodness to the selection algorithm. However, without a suitable *stopping criterion* the FS process may run exhaustively or forever through the space of subsets. Stopping criteria can be: (i) whether addition (or deletion) of any feature does not produce a better subset; and (ii) whether an optimal subset according to some evaluation function is obtained.

Ideally, feature selection methods search through the subsets of features, and try to find the best one among the competing 2^N candidate subsets according to some evaluation function. However, this procedure is exhaustive as it tries to find only the best one. It may be too costly and practically prohibitive, even for a medium-sized feature set size (N). Other methods based on heuristic or random search methods attempt to reduce computational complexity by compromising performance.

Langley [28] grouped different FS methods into two broad groups (*i.e.*, filter and wrapper) based on their dependence on the inductive algorithm that will finally use the selected subset. *Filter* methods are independent of the inductive algorithm, whereas *wrapper* methods use the inductive algorithm as the evaluation function. The filter evaluation functions can be divided into four categories: *distance, information, dependence* and *consistency*.

- *Distance*: For a two-class problem, a feature *X* is preferred to another feature *Y* if *X* induces a greater difference between the two-class conditional probabilities than *Y* [24].
- *Information:* Feature *X* is preferred to feature *Y* if the information gain from feature *X* is greater than that from feature *Y* [7].
- *Dependence:* The coefficient is a classical dependence measure and can be used to find the correlation between a feature and a class. If the correlation of feature *X* with class *C* is higher than the correlation of feature *Y* with *C*, then feature *X* is preferred to *Y* [20].
- *Consistency:* two samples are in conflict if they have the same values for a subset of features but disagree in the class they represent [31].

Relief [24] uses a statistical method to select the relevant features. Relief randomly picks a sample of instances and for each instance in it finds *Near Hit* and *Near Miss* instances based on the Euclidean distance measure. *Near Hit* is the instance having minimum Euclidean distance among all instances of the same class as that of the chosen instance; *Near Miss* is the instance having minimum Euclidean distance among all instances of different class. It updates the weights of the features that are initialized to zero in the beginning based on an intuitive idea that a feature is more relevant if it distinguishes between an instance and its *Near Miss* and less relevant if it distinguishes between an instance and its *Near Hit*. After exhausting all instances in the sample, it chooses all features having weight greater than or equal to a threshold.

Several researchers have explored the possibility of using a particular learning algorithm as a pre-processor to discover useful feature subsets for a primary learning algorithm. Cardie [7] describes the application of decision tree algorithms to the task of selecting feature subsets for use by instance based learners. C4.5 is run over the training set and the features that appear in the pruned decision tree are selected. In a similar approach, Singh and Provan [45] use a greedy *oblivious* decision tree algorithm to select features from which to construct a Bayesian network. BDSFS (Boosted Decision Stump FS) uses boosted decision stumps as the pre-processor to discover the feature subset [9]. The number of features to be selected is a parameter, say *k*, to the FS algorithm. The boosting algorithm is run for *k* rounds, and at each round all features that have previously been selected are ignored. If a feature is selected in any round, it becomes part of the set that will be returned.

FS with neural nets can be thought of as a special case of architecture pruning, where input features are pruned, rather than hidden neurons or weights. The neural-network feature selector (NNFS) is based on elimination of input layer weights [44]. The weights-based feature saliency measures bank on the idea that weights connected to important features attain large absolute values while weights connected to unimportant features would probably attain values somewhere near zero.

Some of FS procedures are based on making comparisons between the saliency of a candidate feature and the saliency of a noise feature [1].

LVF [31] is consistency driven method can handle noisy domains if the approximate noise level is known a-priori. LVF generates a random subset *S* from the feature subset space during each round of execution. If *S* contains fewer features than the current best subset, the *inconsistency* rate of the dimensionally reduced data described by *S* is compared with the *inconsistency* rate of the best subset. If *S* is at least as consistent as the best subset, replaces the best subset. LVS [32] is a variance of LVF that can decrease the number of checkings especially in the case of large datasets.

In Hall [17], a correlation measure is applied to evaluate the goodness of feature subsets based on the hypothesis that a good feature subset is one that contains features highly correlated with the class, yet uncorrelated with each other. Yu and Liu [50] introduced a novel concept, predominant correlation, and proposed a fast filter method which can identify relevant features as well as redundancy among relevant features without pairwise correlation analysis.

Generally, an optimal subset is always relative to a certain evaluation function (*i.e.*, an optimal subset chosen using one evaluation function may not be the same as that which uses another evaluation function). Wrapper methods wrap the FS around the induction algorithm to be used, using cross-validation to predict the benefits of adding or removing a feature from the feature subset used. In forward stepwise selection, a feature subset is iteratively built up. Each of the unused variables is added to the model in turn, and the variable that most improves the model is selected. In backward stepwise selection, the algorithm starts by building a model that includes all available input variables (i.e. all bits are set). In each iteration, the algorithm locates the variable that, if removed, most improves the performance (or causes least deterioration). A problem with forward selection is that it may fail to include variables that are interdependent, as it adds variables one at a time. However, it may locate small effective subsets quite rapidly, as the early evaluations, involving relatively few variables, are fast. In contrast, in backward selection interdependencies are well handled, but early evaluations are relatively expensive. Due to the naive Bayes classifier's assumption that, within each class, probability distributions for features are independent of each other, Langley and Sage [29] note that its performance on domains with redundant features can be improved by removing such features. A forward search strategy is employed to select features for use with naïve Bayes, as opposed to the backward strategies that are used most often with decision tree algorithms and instance based learners.

Sequential forward floating selection (SFFS) and sequential backward floating selection (SBFS) are characterized by the changing number of features included or eliminated at different stages of the procedure [46]. The adaptive floating search [47] is able to find a better solution, of course at the expense of significantly increased computational time.

The genetic algorithm is another well-known approach for FS [49]. In each iteration, a feature is chosen and raced between being in the subset or excluded from it. All combinations of unknown features are used with equal probability. Due to the probabilistic nature of the search, a feature that should be in the subset will win the race, even if it is dependent on another feature. An important aspect of the genetic algorithm is that it is explicitly designed to exploit epistasis (that is, interdependencies between bits in the string), and thus should be well-suited for this problem domain. However, genetic algorithms typically require a large number of evaluations to reach a minimum.

Piramuthu [39] compared a number of FS techniques without finding a real winner. To combine the advantages of filter and wrapper models, algorithms in a hybrid model have recently been proposed to deal with high dimensional data [25]. In these algorithms, first, a goodness measure of feature subsets based on data characteristics is used to choose best subsets for a given cardinality, and then, cross validation is exploited to decide a final best subset across different cardinalities.

VII. Feature construction

The problem of *feature interaction* can be also addressed by constructing new features from the basic feature set. This technique is called *feature construction/transformation*. The new generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to better comprehensibility of the produced classifier, and better understanding of the learned concept.

Assuming the original set *A* of features consists of $a_1, a_2, ..., a_n$, some variants of feature transformation is defined below. Feature transformation process can augment the space of features by inferring or creating additional features. After feature construction, we may have additional *m* features $a_{n+1}, a_{n+2}, ..., a_{n+m}$. For example, a new feature a_k ($n < k \leq n + m$) could be constructed by performing a logical operation of a_i and a_j from the original set of features.

The *GALA* algorithm [19] performs feature construction throughout the course of building a decision tree classifier. New features are constructed at each created tree node by performing a branch and bound search in feature space. The search is performed by iteratively combining the feature having the highest *InfoGain* value with an original basic feature that meets a certain filter criterion. *GALA* constructs new binary features by using logical operators such as conjunction, negation, and disjunction. On the other hand, Zheng [51] creates at-least *M*-of-*N* features. For a given instance, the value of an at-least *M*-of-*N* representation is true if at least *M* of its conditions is true of the instance while it is false, otherwise.

Feature transformation process can also extract a set of new features from the original features through some functional mapping. After feature extraction, we have $b_1, b_2,..., b_m$ ($m < n$), $b_i = f_i(a_1, a_2,...,a_n)$, and f_i is a mapping function. For instance for real valued features a_1 and a_2, for every object x we can define $b_1(x) = c_1*a_1(x) + c_2*a_2(x)$ where c_1 and c_2 are constants. While FICUS [36] is similar in some aspects to some of the existing feature construction algorithms (such as GALA), its main strength and contribution are its generality

and flexibility. FICUS was designed to perform feature generation given any feature representation specification (mainly the set of constructor functions) using its general-purpose grammar.

The choice between FS and feature construction depends on the application domain and the specific training data, which are available. FS leads to savings in measurements cost since some of the features are discarded and the selected features retain their original physical interpretation. In addition, the retained features may be important for understanding the physical process that generates the patterns. On the other hand, transformed features generated by feature construction may provide a better discriminative ability than the best subset of given features, but these new features may not have a clear physical meaning.

VIII. Conclusion

Machine learning algorithms automatically extract knowledge from machine-readable information. Unfortunately, their success is usually dependant on the quality of the data that they operate on. If the data is inadequate, or contains extraneous and irrelevant information, machine learning algorithms may produce less accurate and less understandable results, or may fail to discover anything of use at all. Thus, data pre-processing is an important step in the machine learning process. The pre-processing step is necessary to resolve several types of problems include noisy data, redundancy data, missing data values, etc. All the inductive learning algorithms rely heavily on the product of this stage, which is the final training set.

By selecting relevant instances, experts can usually remove irrelevant ones as well as noise and/or redundant data. The high quality data will lead to high quality results and reduced costs for data mining. In addition, when a data set is too huge, it may not be possible to run a ML algorithm. In this case, instance selection reduces data and enables a ML algorithm to function and work effectively with huge data.

In most cases, missing data should be pre-processed so as to allow the whole data set to be processed by a supervised ML algorithm. Moreover, most of the existing ML algorithms are able to extract knowledge from data set that store discrete features. If the features are continuous, the algorithms can be integrated with a discretization algorithm that transforms them into discrete attributes. A number of studies [23], [20] comparing the effects of using various discretization techniques (on common ML domains and algorithms) have found the entropy based methods to be superior overall.

Feature subset selection is the process of identifying and removing as much of the irrelevant and redundant information as possible. Feature wrappers often achieve better results than filters due to the fact that they are tuned to the specific interaction between an induction algorithm and its training data. However, they are much slower than feature filters. Moreover, the problem of feature interaction can be addressed by constructing new features from the basic feature set (feature construction). Generally, transformed features generated by feature construction may provide a better discriminative ability than the best subset of given features, but these new features may not have a clear physical meaning.

It would be nice if a single sequence of data pre-processing algorithms had the best performance for each data set but this is not happened. Thus, we presented the most well known algorithms for each step of data pre-processing so that one achieves the best performance for their data set.

References

[1] Bauer, K.W., Alsing, S.G., Greene, K.A., 2000. Feature screening using signal-to-noise ratios. Neurocomputing 31, 29–44.

[2] M. Boulle. Khiops: A Statistical Discretization Method of Continuous Attributes. Machine Learning 55:1 (2004) 53-69

[3] Breunig M. M., Kriegel H.-P., Ng R. T., Sander J.: 'LOF: Identifying Density-Based Local Outliers', Proc. ACM SIGMOD Int. Conf. On Management of Data (SIGMOD 2000), Dallas, TX, 2000, pp. 93-104.

[4] Brodley, C.E. and Friedl, M.A. (1999) "Identifying Mislabeled Training Data", AIR, Volume 11, pages 131-167.

[5] Bruha and F. Franek: Comparison of various routines for unknown attribute value processing: covering paradigm. International Journal of Pattern Recognition and Artificial Intelligence, 10, 8 (1996), 939-955

[6] J.R. Cano, F. Herrera, M. Lozano. Strategies for Scaling Up Evolutionary Instance Reduction Algorithms for Data Mining. In: L.C. Jain, A. Ghosh (Eds.) Evolutionary Computation in Data Mining, Springer, 2005, 21-39

[7] C. Cardie. Using decision trees to improve cased-based learning. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1995.

[8] M. Dash, H. Liu, Feature Selection for Classification, Intelligent Data Analysis 1 (1997) 131–156.

[9] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. Proc. of the 8th International Conference on Machine Learning, 2001.

[10] T. Elomaa, J. Rousu. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. Data Mining and Knowledge Discovery 8:2 (2004) 97-126

[11] Fayyad U., and Irani K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In Proc. of the Thirteenth Int. Joint Conference on Artificial Intelligence, 1022-1027.

[12] Friedman, J.H. 1997. Data mining and statistics: What's the connection? Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics.

[13] Marek Grochowski, Norbert Jankowski: Comparison of Instance Selection Algorithms II. Results and Comments. ICAISC 2004a: 580-585.

[14] Jerzy W. Grzymala-Busse and Ming Hu, A Comparison of Several Approaches to Missing Attribute Values in Data Mining, LNAI 2005, pp. 378−385, 2001.

[15] Isabelle Guyon, André Elisseeff; An Introduction to Variable and Feature Selection, JMLR Special Issue on Variable and Feature Selection, 3(Mar):1157--1182, 2003.

[16] Hernandez, M.A.; Stolfo, S.J.: Real-World Data is Dirty: Data Cleansing and the Merge/Purge Problem. Data Mining and Knowledge Discovery 2(1):9-37, 1998.

[17] Hall, M. (2000). Correlation-based feature selection for discrete and numeric class machine learning. Proceedings of the Seventeenth International Conference on Machine Learning (pp. 359–366).

[18] K. M. Ho, and P. D. Scott. Reducing Decision Tree Fragmentation Through Attribute Value Grouping: A Comparative Study, in Intelligent Data Analysis Journal, 4(1), pp.1-20, 2000.

[19] Hu, Y.-J., & Kibler, D. (1996). Generation of attributes for learning algorithms. Proc. 13th International Conference on Machine Learning.

[20] J. Hua, Z. Xiong, J. Lowey, E. Suh, E.R. Dougherty. Optimal number of features as a function of sample size for various classification rules. Bioinformatics 21 (2005) 1509-1515

[21] Norbert Jankowski, Marek Grochowski: Comparison of Instances Selection Algorithms I. Algorithms Survey. ICAISC 2004b: 598-603.

[22] Knorr E. M., Ng R. T.: 'A Unified Notion of Outliers: Properties and Computation', Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), Newport Beach, CA, 1997, pp. 219-222.

[23] R. Kohavi and M. Sahami. Error-based and entropy-based discretisation of continuous features. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1996.
[24] Kononenko, I., Simec, E., and Robnik-Sikonja, M.(1997).Overcoming the myopia of inductive learning algorithms with RELIEFF. Applied Intelligence, 7: 39–55.
[25] S. B. Kotsiantis, P. E. Pintelas (2004), Hybrid Feature Selection instead of Ensembles of Classifiers in Medical Decision Support, Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems, July 4-9, Perugia - Italy, pp. 269-276.
[26] Kubat, M. and Matwin, S., 'Addressing the Curse of Imbalanced Data Sets: One Sided Sampling', in the Proceedings of the Fourteenth International Conference on Machine Learning, pp. 179-186, 1997.
[27] Lakshminarayan K., S. Harp & T. Samad, Imputation of Missing Data in Industrial Databases, Applied Intelligence 11, 259–275 (1999).
[28] Langley, P., Selection of relevant features in machine learning. In: Proceedings of the AAAI Fall Symposium on Relevance, 1–5, 1994.
[29] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In Proc. of 10th Conference on Uncertainty in Artificial Intelligence, Seattle, 1994.
[30] Ling, C. and Li, C., 'Data Mining for Direct Marketing: Problems and Solutions', Proceedings of KDD-98.
[31] Liu, H. and Setiono, R., A probabilistic approach to feature selection—a filter solution. Proc. of International Conference on ML, 319–327, 1996.
[32] H. Liu and R. Setiono. Some Issues on scalable feature selection. Expert Systems and Applications, 15 (1998) 333-339. Pergamon.
[33] Liu, H. and H. Metoda (Eds), Instance Selection and Constructive Data Mining, Kluwer, Boston, MA, 2001
[34] H. Liu, F. Hussain, C. Lim, M. Dash. Discretization: An Enabling Technique. Data Mining and Knowledge Discovery 6:4 (2002) 393-423.
[35] Maas W. (1994). Efficient agnostic PAC-learning with simple hypotheses. Proc. of the 7th ACM Conf. on Computational Learning Theory, 67-75.
[36] Markovitch S. & Rosenstein D. (2002), Feature Generation Using General Constructor Functions, Machine Learning, 49, 59–98, 2002.
[37] Oates, T. and Jensen, D. 1997. The effects of training set size on decision tree complexity. In ML: Proc. of the 14th Intern. Conf., pp. 254–262.
[38] Pfahringer B. (1995). Compression-based discretization of continuous attributes. Proc. of the 12th International Conference on Machine Learning.
[39] S. Piramuthu. Evaluating feature selection methods for learning in data mining applications. European Journal of Operational Research 156:2 (2004) 483-494
[40] Pyle, D., 1999. Data Preparation for Data Mining. Morgan Kaufmann Publishers, Los Altos, CA.
[41] Quinlan J.R. (1993), C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, California.
[42] Reinartz T., A Unifying View on Instance Selection, Data Mining and Knowledge Discovery, 6, 191–210, 2002, Kluwer Academic Publishers.
[43] Rocke, D. M. and Woodruff, D. L. (1996) "Identification of Outliers in Multivariate Data," Journal of the American Statistical Association, 91, 1047–1061.
[44] Setiono, R., Liu, H., 1997. Neural-network feature selector. IEEE Trans. Neural Networks 8 (3), 654–662.
[45] M. Singh and G. M. Provan. Efficient learning of selective Bayesian network classifiers. In Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, 1996.
[46] Somol, P., Pudil, P., Novovicova, J., Paclik, P., 1999. Adaptive floating search methods in feature selection. Pattern Recognition Lett. 20 (11/13), 1157–1163.
[47] P. Somol, P. Pudil. Feature Selection Toolbox. Pattern Recognition 35 (2002) 2749-2759.
[48] C. M. Teng. Correcting noisy data. In Proc. 16th International Conf. on Machine Learning, pages 239–248. San Francisco, 1999.
[49] Yang J, Honavar V. Feature subset selection using a genetic algorithm. IEEE Int Systems and their Applications 1998; 13(2): 44–49.
[50] Yu and Liu (2003), Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC.
[51] Zheng (2000), Constructing X-of-N Attributes for Decision Tree Learning, Machine Learning, 40, 35–75, 2000, Kluwer Academic Publishers.

S. B. Kotsiantis received a diploma in mathematics, a Master and a Ph.D. degree in computer science from the University of Patras, Greece. His research interests are in the field of data mining and machine learning. He has more than 40 publications to his credit in international journals and conferences.

D. Kanellopoulos received a diploma in electrical engineering and a Ph.D. degree in electrical and computer engineering from the University of Patras, Greece. He has more than 40 publications to his credit in international journals and conferences.

P. E. Pintelas is a Professor in the Department of Mathematics, University of Patras, Greece. His research interests are in the field of educational software and machine learning. He has more than 100 publications in international journals and conferences.

