UDC: 007.52
Original scientific paper

A TOOL FOR SUPPORT OF THE KDD PROCESS

Jan Paralič, Peter Bednar
Department of Cybernetics and Artificial Intelligence, University of Technology Košice,
Košice, Slovak Republic
Jan.Paralic@tuke.sk

Abstract: *This paper presents basic ideas and results of the GOAL[1] project focusing on its knowledge discovery part. Within this project a KDD (Knowledge Discovery in Databases) package [7] has been designed and implemented. In this paper motivation, architecture, functionality and one two of the implemented DM (data mining) modules of the KDD Package are described in greater detail. KDD Package supports the whole KDD process [10] starting with possibility to connect to various data sources, following with support for data preprocessing, data mining and knowledge visualization. Based on the GOAL pilot applications one of the DM tasks that are supported now, is the classification DM task (by means of C4.5, CN2, RISE and two-way induction approaches). KDD Package is designed as an open system with easy integration of new DM, data preprocessing or visualization modules.*

Keywords: *knowledge discovery, data mining, databases, knowledge visualization, GIS.*

1. INTRODUCTION

The integration and combination of GIS (Geographic Information System) data into and with OLAP (On-Line Analytical Processing) systems poses a number of yet not satisfyingly solved problems in terms of getting the data into the OLAP system [11], representing the data for analysis and extracting knowledge while considering security restrictions. Current approaches do not address these special problems resulting from the targeted application arena of GIS and OLAP systems.

The main objective of the GOAL project is to develop a generic framework both recognized by the research community and applicable in real world applications, which solves the general issues of GIS and DWH (Data Warehouse) interoperability [8], including DWH feeding [9], knowledge extraction, interpretation, and security concepts. The feasibility of this framework is being tested on 2 very different real world applications from the GIS domain using environmental sensor data of a water supply company and cultural data about historical monument visitors, allowing a real world evaluation of the framework.

Within this project a KDD (Knowledge Discovery in Databases) package [7] has been designed and implemented at the Department of Cybernetics and Artificial Intelligence, University of Technology in Kosice.

In this paper the KDD Package is presented in greater detail. Starting with motivation for development of such a system in the context of the GOAL project mentioned above

[1] International Copernicus research project No. 977091 Geographic Information On-Line Analysis (GIS – Data Warehouse Integration) supported by the European Commission.

(Sect. 2) there is further presented architecture of the KDD Package in Section 3. Sections 4 and 5 provide a detailed view of the most interesting part of the system, namely DM modules supporting classification task. Finally, Section 6 concludes the paper with a summary of the main ideas presented here.

2. MOTIVATION

Techniques provided by the OLAP systems enable to analyze data from data warehouses in a quite easy way, but they are lacking algorithms for more sophisticated analysis of data provided by the KDD approach. Therefore one of the three major objectives of the GOAL project is to develop a KDD supporting tool which shall be integrated with the generic framework of the GIS – DWS integration system [9]. Moreover, DM algorithms implemented within the KDD Package prototype should effectively support two real pilot applications mentioned in Sect. 1.

In the following KDD process will be briefly introduced as a basic reference for KDD Package design objectives presented afterwards.

2.1. KDD PROCESS

Knowledge discovery **in databases** (KDD) can be defined as nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. According to [4] it is an interactive and iterative process with several steps. It means that at any stage the user should have possibility to make changes (for instance to choose different algorithm settings, different DM task or preprocess data in another way) and repeat the following steps to achieve better results. Data mining is a part of this process.

In most of sources, the term **Data Mining** (DM) is often used to name the field of knowledge discovery. This confusing use of terms KDD and DM is due to historical reasons and due to fact that the most of the work is focused on refinement and applicability experiments of machine learning algorithms from artificial intelligence for the data-mining step. Pre-processing is often included in this step as a part of mining algorithm.

Within the KDD process following steps can be recognized [5], [10].

1. *Data cleaning* to remove noise and inconsistent data
2. *Data integration*, where multiple data sources may be combined
3. *Data selection,* where data relevant to the analysis task are retrieved from database (or data warehouse, where data is already cleaned and integrated)
4. *Data transformation* - data are transformed or consolidated into forms appropriate for mining
5. *Data mining* (DM) [15] as core of the KDD process, where intelligent methods are applied in order to extract data patterns
6. *Pattern evaluation* – to identify interesting patterns
7. *Knowledge representation* - visualization of mined knowledge.

2.2. MAIN DESIGN OBJECTIVES OF THE KDD PACKAGE

Software that aims effectively support the KDD process cannot focus just on one of the above-mentioned steps but should cover ideally all of them. Therefore in order to adopt existing data mining algorithms to be used in connection with real data sources (in a database or in a data warehouse), four crucial objectives have been identified for KDD Package design.

1. Fast connection to existing data sources. In case of the GOAL project not only databases, but also data warehouses must be taken into account.
2. Flexible and rich set of data preprocessing methods (which involves steps 1 to 4 in KDD process presented in Sect. 2.1 above) must be provided (in form, which is easy to use and easy to understand by the user).
3. The system must be open for easy integration of new data mining algorithms (step 5).
4. Effective support of evaluation of discovered patterns must be provided by means of (preferably) visual knowledge representation (steps 6 and 7).

3. ARCHITECTURE

Based on objectives presented in Sect. 2.2 above, KDD package has been design [12] and implemented in such a way, that it can be used not only within the GOAL generic framework, but also as a stand-alone application supporting the whole KDD process.

KDD Package has modular structure (see Figure 1), where common parts of the system can be used by each of the specialized DM modules.

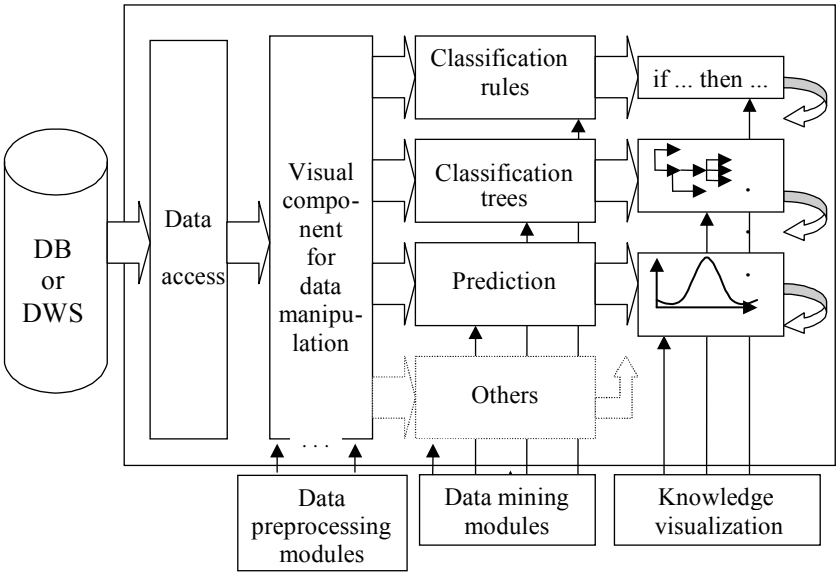

Figure 1: Architecture of the KDD Package.

Data access module serves for accessing database sources of various types (which can be a text file, DBase, Paradox or MS Excel table, any local or remote SQL database using a visual SQL wizard or data warehouse through MS Excel interface). Data is connected (basic statistical measures are provided at the same time. If user decides to process actually connected data, it will be loaded) and forms a so called *view*, which can be previewed in three levels of granularity – list of *operations* performed on actual view, list of attributes with statistics (so called *quick view*) or a two dimensional table of *data* (see Fig. 2).

Module for data pre-processing enables a user to visualize, browse, modify, transform, sample etc. connected data. Pre-processing techniques are divided into those, which operate on rows, and those, which operate on columns of a view (see Fig. 2).

All possible operations on data are defined by means of plug-in modules. This makes it possible to add a new transformation (sampling, discretization, etc.) operation on data any time very easily.

For each KDD task a different DM algorithm as well as knowledge visualization component is suitable. Therefore each new DM algorithm and its respective knowledge visualization component can be implemented separately and added into the KDD package in form of a separate plug-in module. It does not necessarily mean that each DM algorithm must have its own knowledge visualization component.

Usually in order to add a new DM task functionality into the KDD Package the following steps need to be done.

1. Implementation (or just re-use of an existing implementation) of a data-mining algorithm in form of a plug-in module
2. If necessary, implementation of new transformation or other pre-processing functions in form of plug-in module
3. If necessary, implementation of a new knowledge visualization component in form of a plug-in module

Process of adding a new plug-in module may be controlled directly from the running KDD package application.

Based on character of the real data from the GOAL project pilot applications, classification and prediction DM tasks functionality have been implemented by means of various algorithms and their combination (see Sect. 4 and 5). KDD Package with its DM modules is being tested on the real data from two pilot applications as well as on the data from UCI repository.

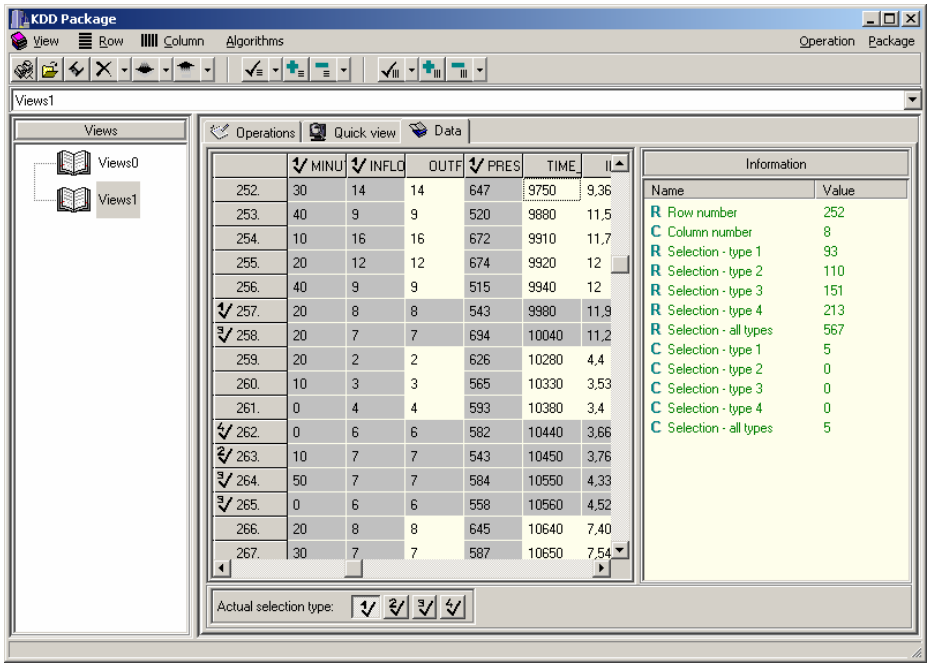

Figure 2: Main window of the KDD Package with two opened views (data sources). The second view is currently selected as active and presented at the finest level of granularity (data). There are some rows and columns selected for further processing using 4 different selection types.

4. CLASSIFICATION

This is particularly useful for pilot application dealing with ticket sales data from historical monuments in South Bohemia [14]. There are more algorithms available in KDD package with two different knowledge visualization components.

1. *Classification trees* are produced by means of C4.5 algorithm [13] (see Figure 3)
2. *Classification rules* are produced by means of CN2 and RISE (see Section 4.1 and Figure 4)

Moreover, four algorithms that combine decision rules learned by general-to-specific learners, like CN2 or C4.5 on one hand side (based on "divide and conquer" strategy) and the RISE algorithm that performs specific-to-general induction on the other side have been implemented and will be presented in Sect. 4.2.

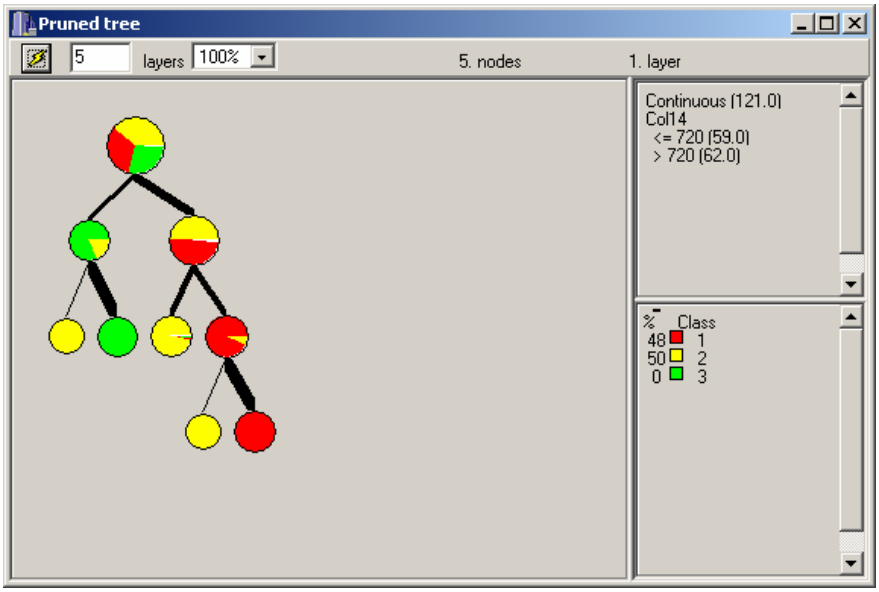

Figure 3: Graphical representation of a decision tree produced by the C4.5 implementation within the KDD package.

4.1 RULE INDUCTION AND INSTANCE-BASED LEARNING

Rule induction algorithms have several advantages. Most notable one (especially for KDD) is that rules are the most easily understood by humans. Other advantages of rule induction algorithms include their ability to select relevant attributes in high-dimensional instance spaces, their natural suitability for symbolic domains, and ability to easily mix symbolic and numeric attributes. Because only statistical measures are used to evaluate induction, good noise immunity can by achieved.

Rule induction algorithms like CN2 [1] employ "divide and conquer" search strategy. In this strategy algorithm forms a class definition by constructing a rule that covers many positive examples, and few negatives, then separates out the covered examples and continues again on the remainder only. This splintering of the training set may cause later rules, and later conditions within each rule, to be induced with insufficient statistical support (sample of training examples), leading to greater noise sensitivity and missing or incorrect rules or conditions. This problem is denoted as a fragmentation problem.

Another problem of rule induction algorithms is *small disjuncts problem* [6]: rules covering few training examples tend to be highly error sensitive, but removing them decreases the global accuracy. Some of these small disjuncts correspond to rare cases, and are difficult to learn, because only a small sample set of such cases is available. However, other small disjuncts may be a product of the fragmentation problem.

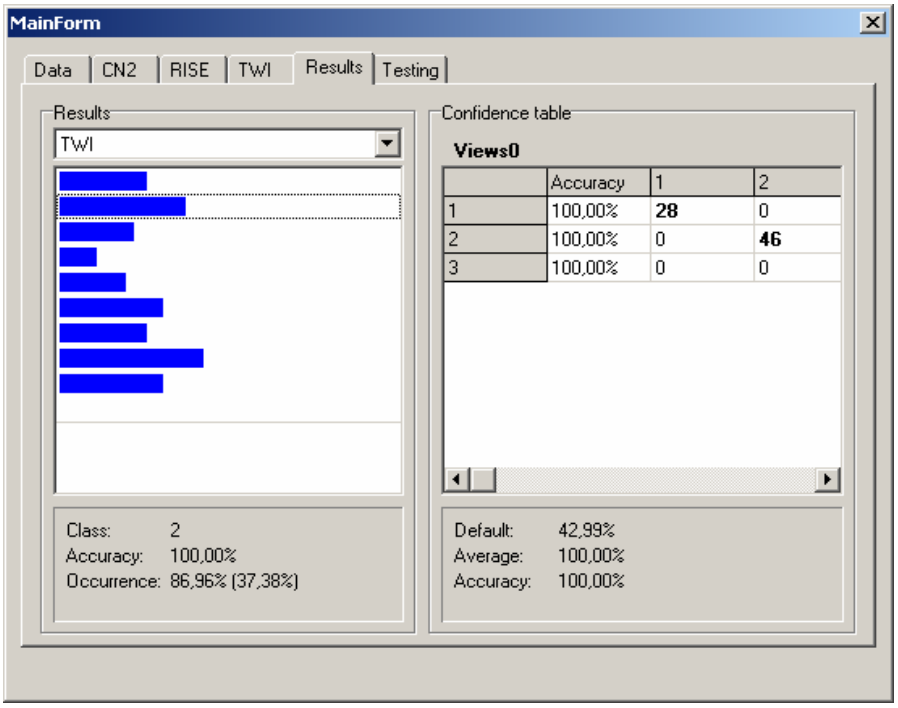

Figure 4: Visual representation of decision rules produced by the rule induction algorithms within the KDD package.

The RISE [2] algorithm reduces some of the introduced problems of the rule induction algorithms. This algorithm unifies rule induction and instance-based learning (IBL). Instance-based learners are able to form complex decision boundary in the instance space. Special cases that may be missed by abstraction forming algorithm can be retained and recognized. IBL have some disadvantages, however. The memory cost of the class descriptions they produce is typically greater, and they can be harder for a human to understand. Most significant problem for IBL is arguably that posed by irrelevant attributes. The contribution of these attributes to the global distance constitutes the noise and they can swamp out the relevant component.

Another problem is that some attributes may be relevant only in context (i.e., given the values of other attributes). However, RISE algorithm is able to selectively generalize and drop attributes from instances, and thus to reduce the context-dependency problems that can affect instance based learners. RISE employs a specific-to-general; "conquering without separating" search strategy. This allows it to reduce the fragmentation and small disjuncts problems.

4.2 TWO-WAY INDUCTION

When the target description is a mix of general rules and more specific "exceptions" areas, both approaches can have problems to find appropriate description. General-to-specific learners may not recognize the exception areas, and specific-to-general ones may induce only imperfect, corrupted general rules. A natural solution would be to combine the

two search directions in a single system. In the following such approaches used within the KDD Package will be presented.

In proposed system CN2 is used as the general-to-specific learner and the RISE algorithm performs specific-to-general induction. Let *S* is a set of rules produced by the RISE algorithm applied to a training set and *G* is a set of rules resulted by application of a general-to-specific learner (in case of KDD Package the CN2 algorithm).

TWI-1 algorithm. In the TWI-1 algorithm [3], the sets of *S* and *G* rules are merged to form a single set, deleting any duplicates. The Laplace accuracy of each rule on the examples covered by this rule is measured and the classification procedure of the RISE algorithm is applied to each training example. Each rule memorizes the number of examples it won, and how many of them it classifies correctly. At the end, the Laplace accuracy of each rule on the examples that it won is computed. At classification time the RISE classification procedure is applied without any distinction between *S* and *G* rules. The nearest rule to the test example wins, if two or more rules are equally near, the one with highest Laplace accuracy prevails.

TWI-2 algorithm. In the TWI-2 algorithm [3], the sets of *S* and *G* rules are kept separate, and the Laplace accuracy of each rule on the training examples that it covers is measured. At classification time, the two sets of rules are first applied separately. A winner among the *S* rules is found by the RISE classification procedure. To select the *G* winner, the *G* rules are matched against the test example. If more than one rule covers this example, the one with the highest Laplace accuracy wins. If no *G* rule covers this example, the default rule is chosen as the *G* winner. From such two rules (*S* winner and *G* winner) the one with higher accuracy wins the example.

In addition to two above-mentioned approaches proposed by P. Domingos, we proposed and implemented two other approaches, which we call TWI-ID3 and TWI-BOOST respectively.

TWI-ID3 algorithm. The TWI-2 algorithm is based on *stacked generalization* architecture [16]. This architecture consists of component classifiers that are used in combination. Classifier to be combined is denoted as *level-0* classifier, and the combining classifier is the *level-1* classifier. Stacked generalization architecture may be generalized more levels of classifiers. Each layer of the classifiers is used to combine the predictions of the classifiers in the layer immediately below. In TWI-2 algorithm, combining algorithm is voting (between two finalists, *S* winner and *G* winner). We proposed the TWI-ID3 algorithm that combines classifier decision tree generated by ID3 [13] algorithm. The task of the *level-1* (and higher) classifier is to learn to use the contestant predictions to predict more accurately. In this case decision tree classifier can be more sophisticated method to combine *level-0* classifiers. Decision tree is constructed by means of ID3 algorithm from a set of examples (number of examples is the same as in original set) with *n+1* nominal attributes (*n* is the number of classifiers used in *level-0*, in our case two, i.e. CN2 and RISE). Value of *i*-th attribute for particular example represents class predicted for that example by *i*-th classifier. The last attribute is the real class of that example.

TWI-BOOST algorithm. If we have given base classifier and we want to increase the generalization accuracy with *stacked generalization* architecture, accuracy and diversity are two most important criteria for component classifiers. Combining the predictions of a set of classifiers that all make the same errors cannot lead to any improvement in the accuracy of the composite prediction. The TWI-BOOST algorithm is designed to create a diverse component classifier learned by RISE algorithm for a set of decision rules learned by CN2 algorithm. At first algorithm divides training set into two subsets. On first subset, a set of CN2 decision rules (*G* set of rules) is learned. This set of rules is than applied to second training subset. The RISE algorithm than generalizes rules constructed only from incorrectly classified examples from this second training subset (constructing *S* set of rules). TWI-1, TWI-2, or TWI-ID3 algorithms can combine final S and G sets of rules.

5 EXPERIMENTS

In first part of our experiments 12 databases from well-known UCI repository of standard benchmark datasets have been used. The results of our experiments are compared on three different parameters measured, namely: 1) algorithms accuracy, 2) complexity of discovered knowledge (in form of learned list of rules) and 3) learning efficiency.

In order to have better view how different types of attributes influence achieved results, we used 5 databases with nominal attributes, 4 with real attributes and 3 databases with mixed type of attributes have been used for testing. Data in each test was randomly splitted into 2/3 for training and 1/3 for testing. Table 1 shows the estimated accuracy for random classification for each database, number of classes, number of attributes and number of training and testing examples for each database.

Table 1: Databases from UCI repository used for testing

Database	*Training examples*	*Testing examples*	*Attributes* [N, R]	*Classes*	*Accuracy* [%]
Echocardiogram	87	44	0, 7	2	66,67
Glass	142	72	0, 9	6	35,14
Hayes-roth[1]	106	53	4, 0	3	40
Heart-desease[2]	202	101	8, 5	2	53,92
Hepatitis	103	52	13, 6	2	78,85
Iris	100	50	0, 4	3	33,33
Mushroom	5418	2706	22, 0	2	51,79
Soybean[1]	455	228	35, 0	19	13,19
Thyroid-desease[1]	2515	1257	22, 7	4	92,21
Tic-tac-toe	638	320	9, 0	2	65,31
Voting-records	290	145	16, 0	2	61,38
Wine	118	60	0, 13	3	40

[1] testing and training datasets was joined and split again

[2] database Cleveland

Table 2 shows accuracy obtained in our tests for two base algorithms CN2 and RISE.

Table 2: Accuracy for base classifiers CN2 and RISE.

Data	*CN2*	*RISE*	*RISE/ CN2*
$Echocardiogram_{20}$	68 ± 6,9	67,1 ± 7,1	0,986
$Glass_{20}$	56,9 ± 5,2	67,9 ± 4,7 $^{+}$	1,193
$Hayes\text{-}roth_{20}$	80,4 ± 4,6	85,3 ± 4,5 $^{+}$	1,06
$Heart\text{-}desease_{20}$	78,6 ± 2,7	79,4 ± 3,7	1,01
$Hepatitis_{20}$	78,8 ± 4,3	77,4 ± 4,8	0,982
$Iris_{20}$	93,6 ± 2,4	93,4 ± 1,8	0,997
$Mushroom_{4}$	100 ± 0	99,9 ± 0	0,999
$Soybean_{20}$	89,7 ± 1,9	90,4 ± 1,6	1,007
$Thyroid\text{-}desease_{2}$	98,2 ± 0,8	98,9 ± 0,1	1,007
$Tic\text{-}tac\text{-}toe_{20}$	93,4 ± 3,4	96,1 ± 1,4	1,028
$Voting\text{-}records_{20}$	95,4 ± 1,4	94,9 ± 2,2	0,994
$Wine_{20}$	92,9 ± 2,4	96,1 ± 2,1 $^{+}$	1,034

$^{+}$ Improvement ≥ 3%

Indices denote number of iterations

For CN2, a star size of 5 was used and significance testing was switched off. Table 3 shows accuracy of algorithms for rules-combining algorithms. Names TWI-1P and TWI-2P denote modifications of TWI-1 and TWI-2 algorithms with pruning. These algorithms remove rules, which do not cover any examples.

Table 3: Accuracy of algorithms for combination of rules.

Data	*TWI-1*	*TWI-1P*	*TWI-2*	*TWI-2P*	*TWI-ID3*
Echocardiogram	65,5 ± 6,5^{-}	64,8 ± 6,0^{-}	69,2 ± 5,4^{+}	70,2 ± 5,3^{+}	67,2 ± 6,9
Glass	61,5 ± 6,2	63,1 ± 6,6	66,3 ± 4,7	63,1 ± 6,6	67,9 ± 4,7
Hayes-roth	80,8 ± 5	81,3 ± 4,7	82,2 ± 6,1	77,2 ± 4,9^{-}	81 ± 4,6
Heart-desease	79,3 ± 2,7	79,3 ± 2,5	79,5 ± 2,8^{+}	79,7 ± 2,9^{+}	78,6 ± 3,6
Hepatitis	80,3 ± 3,7^{+}	80 ± 3,4^{+}	78,6 ± 5,1	78,8 ± 5,2	78,9 ± 4,6^{+}
Iris	92,7 ± 3,4^{-}	93,7 ± 4,7^{+}	93,6 ± 4,7	92,7 ± 6,2^{-}	93,8 ± 4,3^{+}
Mushroom	100 ± 0	100 ± 0	100 ± 0	95,3 ± 3,6^{-}	100 ± 0
Soybean	92 ± 1,4^{+}	92 ± 0,6^{+}	91,2 ± 1,8^{+}	75,8 ± 3,2^{-}	91,2 ± 1,3^{+}
Thyroid-desease	99,3 ± 0,1^{+}	99,3 ± 0,2^{+}	99 ± 0,5^{+}	98,8 ± 0,4	98,7 ± 0,2
Tic-tac-toe	98,4 ± 0,6^{+}	98,5 ± 0,1^{+}	96,4 ± 0,8^{+}	95,2 ± 0,3	97,2 ± 1,1^{+}
Voting-records	95,8 ± 1,3^{+}	95,8 ± 1,2^{+}	95,3 ± 1	95,2 ± 1	96 ± 1,6^{+}
Wine	91,7 ± 2,5^{-}	93,9 ± 2,3	94,7 ± 2,6	94,6 ± 1,6	95,9 ± 0,8

$^{-}$ improvement against base classifiers

$^{+}$ worsening against base classifiers

The number of databases where algorithm TWI-1 is better or worse respectively than base algorithms is 5:3 (for differences higher than 1% is ratio 3:2). Average accuracy for all databases is 86,4%. For TWI-2 accuracy ratio is 5:0, for differences ≥ 1% is 1:0. Average accuracy is 87,1%. For algorithm TWI-ID3 the ratio is 5:0 (1:0 for ≥ 1:0) with average accuracy 87,2%.

Table 4 shows average numbers of rules and conditions for composed classifier after pruning with methods TWI-1P and TWI-2P.

Table 4: Average number of rules and conditions after pruning for algorithms TWI-1P and TWI-2P

Data	*TWI-1P Rules*	*Conditions*	*TWI-2P Rules*	*Conditions*
Echocardiogram	21 [10, 11]	106	23 [11, 12]	106
Glass	28 [6, 22]	216	25 [8, 17]	167
Hayes-roth	19 [19, 0]	46	21 [18, 3]	52
Heart-desease	39 [22, 17]	202	39 [24, 15]	186
Hepatitis	17 [13, 4]	79	20 [13, 7]	111
Iris	7 [3, 4]	21	7 [5, 2]	14
Mushroom	13 [6, 7]	72	16 [10, 6]	74
Soybean	39 [28, 11]	341	42 [31, 11]	400
Thyroid-desease	24 [14, 10]	232	30 [19, 11]	251
Tic-tac-toe	22 [17, 5]	78	23 [20, 3]	75
Voting-records	18 [16, 2]	60	20 [12, 8]	77
Wine	9 [1, 8]	112	8 [4, 4]	65

Number of rules for TWI-2P covers also *default* rule

Next experiment wanted to test boosting algorithm, which is used to improve classification accuracy of given base classifier. As a base classifier CN2 algorithm was used. The RISE algorithm was used to build component classifier. Predictions of these two classifiers are combined with algorithms TWI-1, TWI-2 or TWI-ID3. Table 5 shows classification accuracy for composed classifier.

Table 5: Accuracy of composed classifier for boosting algorithm

Data	*TWI-1*	*TWI-1P*	*TWI-2*	*TWI-2P*	*TWI-ID3*
Echocardiogram	65,5 ± 6,5	64,8 ± 6,0	63,5 ± 5,4	65,2 ± 5,3	66,9 ± 6,9
Glass	56,1 ± 5,2	57,3 ± 6,6	63 ± 5,7	60,6 ± 6,1	62,8 ± 4,7
Hayes-roth	80,4 ± 4,9	80,3 ± 3,1	78,8 ± 4,6	80,3 ± 3,4	81 ± 4,4
Heart-desease	78,9 ± 3,4	79 ± 2,5	79,3 ± 2,8	78,7 ± 4,8	79,6 ± 4,3
Hepatitis	77,7 ± 6	78 ± 5,8	80,6 ± 4,5	80,4 ± 4,4	81,3 ± 4,7
Iris	93,7 ± 3,3	94,1 ± 3,2	93,1 ± 3,8	89,7 ± 7,4	93,8 ± 4,3
Soybean	90,8 ± 1	89,9 ± 1,5	89,7 ± 2,7	85,6 ± 5,2	90,8 ± 2,3
Thyroid-desease	98,5 ± 0,2	98,6 ± 0	96,5 ± 1,4	96 ± 1,5	97,8 ± 0,7
Voting-records	95,2 ± 1,7	95,3 ± 1,2	94 ± 2,2	91,6 ± 3	95,6 ± 2,4
Wine	94,7 ± 3,7	94,8 ± 3,7	92,8 ± 4,4	93,1 ± 4,5	94,1 ± 3,9

Table 6 presents some information about component classifier that was build using RISE algorithm.

Table 6: Information about **component** classifier for boosting algorithm

Data	*Rules*	*TWI-1P Rules*	*TWI-2P Rules*	*Accuracy*	*Generalizations*
Echocardiogram	6	4	3	44,2	0,7. 10^2
Glass	27	9	13	51,6	4,4. 10^2
Hayes-roth	4	1	2	53,1	0,3. 10^2
Heart-desease	10	5	6	66	4. 10^2
Hepatitis	6	3	3	60	1,4. 10^2
Iris	5	2	2	52,4	3,6. 10^2
Soybean	15	4	5	21,1	2. 10^2
Thyroid-desease	23	8	10	88,5	2,9. 10^2
Voting-records	5	1	2	46,2	0,8. 10^2
Wine	3	1	2	50,3	7

According to results, with simple TWI-1 algorithm it is possible to improve classification accuracy against base classifiers. With algorithms TWI-2 and TWI-3 better average accuracy has been obtained, but as a result RISE was the best algorithm according to average accuracy. With pruning modification, TWI-1P is about 0,4% worse than RISE algorithm, but number of rules is significantly lower (only for Glass database, the CN2 algorithm had lower number of rules by 3 rules).

6. SUMMARY

In this paper a tool supporting the whole KDD process called KDD Package has been presented. This tool has been designed and implemented within the GOAL project aiming both effectively support decision making in the GIS-DWS integrated framework as well as provide a stand alone application for any future KDD application.

KDD Package supports the whole KDD process starting with possibility to connect to various data sources like a text file, DBase, Paradox or MS Excel table, any local or remote SQL database or a data warehouse through MS Excel interface. Data preprocessing is supported by means of various row and column operations. Data mining and knowledge visualization provide core steps. Based on the GOAL pilot applications two DM tasks are supported. In this paper, classification DM task has been described in greater details.

Classification may be done using C4.5, CN2, RISE and two-way induction approaches. Two original two-way induction algorithms have been designed and implemented (TWI-ID3 and TWI-BOOST).

KDD Package is designed as an open system with the possibility to easy integrate any new DM task functionality, data preprocessing or knowledge visualization modules.

REFERENCES

[1] Clark, P., Niblett, T.: The CN2 Induction Algorithm. Machine Learning Journal 3 (1989) 261-283

[2] Domingos, P.: The RISE 2.0 System: A Case Study in Multistrategy Learning. Technical Report 95-2, Dept. of Information and Computer Science, Univ. of California, Irvine (1995)

[3] Domingos, P.: A Unified Approach to Concept Learning. Ph.D. thesis, Department of Information and Computer Science, University of California, Irvine (1997)

[4] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data. Comm. of the ACM, 11 (1996), 27–34

[5] Han, J., Kamber, M.: Data Mining – Concepts and Techniques. Morgan Kaufmann Publishers, (2000)

[6] Holte, R.C., Acker, L.E., Porter, B.W.: Concept Learning and the Problem of Small Disjuncts. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, (1989) 813-818

[7] KDD Package with documentation: http://neuron.tuke.sk/~paralic/KDD/

[8] Kouba Z., Mařík V., Mikšovský P., Tjoa A Min: Data Warehousing and Geographical Information, 4th World Multiconference on Systemics, Cybernetics and Informatics - SCI 2000, July 23-26, Orlando, Florida, 2000

[9] Kouba Z., Matoušek K., Mikšovský P., Štěpánková O.: On Data Warehouse Updating from Multiple Data Sources. In. Database and Expert System Applications, Vienna, Springer - Verlag, Heidelberg, (1998) 767-775

[10] Klösgen, W. and Zytkow, J.M. (Eds.): Handbook of Data Mining and Knowledge Discovery. Oxford University Press, (2002)

[11] Kurz A.: Data Warehousing – Enabling Technology, (in German), MITP-Verlag GmbH, Bonn, 1999

[12] Paralič, J., Andrássyová, E.: Intelligent Knowledge Discovery. In: Rudas, I.J., Madarasz, L. (eds.): Proc. from the 3rd IEEE International Conference on Intelligent Engineering Systems INES'99. Elfa Academic Press Ltd., Kosice (1999) 307–310

[13] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California (1993)

[14] Rauber, A., Paralič, J.: Cluster Analysis as a First Step in the Knowledge Discovery Process. Journal of Adv. Comp. Intelligence, 4 (2000)

[15] Witten, I.H., Frank, E.: Data Mining. Morgan Kaufmann Publishers, San Francisco, California (2000)

[16] Wolpert, D.H.: Stacked Generalization. Neural Networks, 5 (1992), 241-259

Received: 25 August 2002
Accepted: 5 September 2003