COMPREHENSIBLE CREDIT-SCORING KNOWLEDGE VISUALIZATION USING DECISION TABLES AND DIAGRAMS

Christophe Mues, Johan Huysmans, Jan Vanthienen
K.U.Leuven
Naamsestraat 69, B-3000 Leuven, Belgium

Bart Baesens
University of Southampton
School of Management, SO17 1BJ Southampton, UK

Keywords: Credit scoring, neural network rule extraction, decision tables, decision diagrams.

Abstract: One of the key decision activities in financial institutions is to assess the credit-worthiness of an applicant for a loan, and thereupon decide whether or not to grant the loan. Many classification methods have been suggested in the credit-scoring literature to distinguish good payers from bad payers. Especially neural networks have received a lot of attention. However, a major drawback is their lack of transparency. While they can achieve a high predictive accuracy rate, the reasoning behind how they reach their decisions is not readily available, which hinders their acceptance by practitioners. Therefore, we have, in earlier work, proposed a two-step process to open the neural network black box which involves: (1) extracting rules from the network; (2) visualizing this rule set using an intuitive graphical representation. In this paper, we will focus on the second step and further investigate the use of two types of representations: decision tables and diagrams. The former are a well-known representation originally used as a programming technique. The latter are a generalization of decision trees taking on the form of a rooted, acyclic digraph instead of a tree, and have mainly been studied and applied by the hardware design community. We will compare both representations in terms of their ability to compactly represent the decision knowledge extracted from two real-life credit-scoring data sets.

1 INTRODUCTION

One of the key decisions financial institutions have to make as part of their daily operations is to decide whether or not to grant a loan to an applicant. With the emergence of large-scale data-storing facilities, huge amounts of data have been stored regarding the repayment behavior of past applicants. It is the aim of credit scoring to analyze this data and build data-mining models that distinguish good applicants from bad applicants using characteristics such as amount on savings account, marital status, purpose of loan, etc. Many machine-learning and statistical techniques have been suggested in the literature to build credit-scoring models (Baesens et al., 2003c; Thomas, 2000). Amongst the most popular are traditional statistical methods (e.g. logistic regression (Steenackers and Goovaerts, 1989)), nonparametric statistical models (e.g. k-nearest neighbor (Henley and Hand, 1997) and classification trees (David et al., 1992)) and neural networks (Baesens et al., 2003b).

However, when looking at today's credit-scoring practice, one typically sees that the estimated classification models, although often based on advanced and powerful algorithms, fail to be successfully integrated into the credit decision environment. One of the key underlying reasons for this problem, is that the extracted knowledge and patterns can not easily be represented in a way that facilitates human interpretation and validation. Hence, properly visualizing the knowledge and patterns extracted by a data-mining algorithm is becoming more and more a critical success factor for the development of decision-support systems for credit scoring.

Therefore, in this paper, we report on the use of different knowledge visualization formalisms for credit scoring. Starting from a set of propositional if-then rules previously extracted by a powerful neural network rule extraction algorithm, we will investigate both decision tables and decision diagrams as alternative knowledge visualization schemes. The latter are a generalization of decision trees taking on the form of a rooted, acyclic digraph instead of a tree, and have mainly been studied and applied by the hardware design community. We will compare both representations in terms of their ability to compactly represent the decision knowledge extracted from two real-life credit-scoring data sets.

Mues C., Huysmans J., Vanthienen J. and Baesens B. (2004).
COMPREHENSIBLE CREDIT-SCORING KNOWLEDGE VISUALIZATION USING DECISION TABLES AND DIAGRAMS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 226-232
DOI: 10.5220/0002598602260232

This paper is organized as follows. Section 2 discusses the basic concepts of decision tables. Section 3 then elaborates on how decision diagrams may provide an alternative, more concise view of the extracted patterns. Empirical results are presented in section 4. Section 5 concludes the paper.

2 DECISION TABLES

Decision tables (DTs) are a tabular representation used to describe and analyze decision situations (e.g. credit-risk evaluation), where the state of a number of conditions jointly determines the execution of a set of actions. A DT consists of four quadrants, separated by double-lines, both horizontally and vertically. The horizontal line divides the table into a condition part (above) and an action part (below). The vertical line separates subjects (left) from entries (right). The condition subjects are the criteria that are relevant to the decision-making process. They represent the attributes of the rule antecedents about which information is needed to classify a given applicant as good or bad. The action subjects describe the possible outcomes of the decision-making process (i.e., the classes of the classification problem: applicant = good or bad). Each condition entry describes a relevant subset of values (called a state) for a given condition subject (attribute), or contains a dash symbol ('-') if its value is irrelevant within the context of that column ('don't care' entry). Subsequently, every action entry holds a value assigned to the corresponding action subject (class). True, false and unknown action values are typically abbreviated by '$\times$', '-', and '.', respectively. Every column in the entry part of the DT thus comprises a classification rule, indicating what action(s) apply to a certain combination of condition states. E.g., in Figure 1 (b), the final column tells us to classify the applicant as good if owns property = no, and savings amount = high.

If each column only contains simple states (no contracted or don't care entries), the table is called an expanded DT, whereas otherwise the table is called a contracted DT. Table contraction can be achieved by combining logically adjacent (groups of) columns that lead to the same action configuration. For ease of legibility, we will allow only contractions that maintain a lexicographical column ordering, i.e., in which the entries at lower rows alternate before the entries above them; see Figure 1 (Figure 2) for an example of an (un)ordered DT, respectively. As a result of this ordering restriction, a decision tree structure emerges in the condition entry part of the DT, which lends itself very well to a top-down evaluation procedure: starting at the first row, and then working one's way down the table by choosing from the relevant condition states, one safely arrives at the prescribed action (class) for a given case. The number of columns in the contracted table can be further minimized by changing the order of the condition rows. It is obvious that a DT with a minimal number of columns is to be preferred since it provides a more parsimonious and comprehensible representation of the extracted knowledge than an expanded DT (see Figure 1).

1. Owns property?	yes				no			
2. Years client	≤ 3		>3		≤ 3		>3	
3. Savings amount	low	high	low	high	low	high	low	high
1. Applicant=good	-	$\times$	$\times$	$\times$	-	$\times$	-	$\times$
2. Applicant=bad	$\times$	-	-	-	$\times$	-	$\times$	-

(a) Expanded DT

1. Owns property?	yes			no	
2. Years client	≤ 3		>3	-	
3. Savings amount	low	high	-	low	high
1. Applicant=good	-	$\times$	$\times$	-	$\times$
2. Applicant=bad	$\times$	-	-	$\times$	-

(b) Contracted DT

1. Savings amount	low			high
2. Owns property?	yes		no	-
3. Years client	≤ 3	>3	-	-
1. Applicant=good	-	$\times$	-	$\times$
2. Applicant=bad	$\times$	-	$\times$	-

(c) Minimum-size contracted DT

Figure 1: Minimizing the number of columns of a lexicographically ordered DT.

1. Savings amount	high	-	low	low
2. Owns property?	-	yes	no	-
3. Years client	-	>3	-	≤ 3
1. Applicant=good	$\times$	$\times$	-	-
2. Applicant=bad	-	-	$\times$	$\times$

Figure 2: Example of an unordered DT.

Note that we deliberately restrict ourselves to single-hit tables, wherein columns have to be mutually exclusive, because of their advantages with respect to verification and validation (Vanthienen et al., 1998). It is this type of DT that can be easily checked for potential anomalies, such as inconsistencies (a particular case being assigned to more than one class) or incompleteness (no class assigned). The decision table formalism thus facilitates the expert's assessment of the knowledge extracted by e.g. a neural network rule extraction algorithm. What's more, consulting a DT in a top-down manner, as suggested above, should prove more intuitive, faster, and less prone to human error, than evaluating a set of rules one by one.

3 DECISION DIAGRAMS

Decision diagrams are a graph-based representation of discrete functions, accompanied by a set of graph algorithms that implement operations on these func-

tions. Given the proper restrictions (cf. infra), decision diagrams have a number of valuable properties:

- they provide a canonical function representation;
- they can be manipulated efficiently;
- for many practically important functions, the corresponding descriptions turn out to be quite compact.

Precisely these properties explain why various types of diagrams have been used successfully in efficiently solving many logic synthesis and verification problems in the hardware design domain. Especially binary decision diagrams (BDDs) have, since the work of Bryant (Bryant, 1986), who defined the canonical subclass of reduced ordered binary decision diagrams, pervaded virtually every subfield in the former areas. There are on the other hand relatively few reported applications so far in the domain of artificial intelligence (Horiyama and Ibaraki, 2002) and machine learning (Kohavi, 1996), while their use for the visual representation of rules extracted from neural networks, or in the application domain of credit scoring, has to our knowledge not been proposed before (note that our approach differs from that presented in (Kohavi, 1996) in that we apply MDDs in a separate visualization step instead of during the learning itself).

Since we are dealing with general discrete (as opposed to binary) attributes, we will apply *multi-valued decision diagrams* (MDDs), a representation similar to BDDs but which does not restrict the outdegree of internal nodes or the number of sink nodes (Kam et al., 1998). An MDD is a rooted, directed acyclic graph, with m sink nodes for every possible output value (class). Each internal node v is labelled by a test variable $var(v) = x_i\,(i = 1, ..., n)$, which can take values from a finite set $range(x_i)$. Each such node v has $\mid range(x_i) \mid$ outgoing edges, and its successor nodes are denoted by $child_k(v)$, for each $k \in range(x_i)$, respectively. An MDD is ordered (OMDD), iff, on all paths through the graph, the test variables respect a given linear order $x_1 < x_2 < ... < x_n$; i.e., for each edge leading from a node labelled by x_i to a node labelled by x_j, it holds that $x_i < x_j$.

An OMDD is meant to represent an n-variable discrete function. For a given assignment to the variables, the function value is determined by tracing a path from the root to a sink, following the edges indicated by the values assigned to the variables. The label of the sink node specifies the function value (class) assigned for that input. Figure 3 displays an example of an OMDD representation for a two-variable function, $\{0, 1, 2, 3\} \times \{0, 1, 2\} \rightarrow \{0, 1\}$, with respect to the variable order $x_1 < x_2$.

Up to here, OMDDs are not yet uniquely determined for each function. However, by further restricting the representation, a canonical form of MDDs is obtained, namely reduced OMDDs (ROMDD). An OMDD is said to be reduced, iff it does not contain a

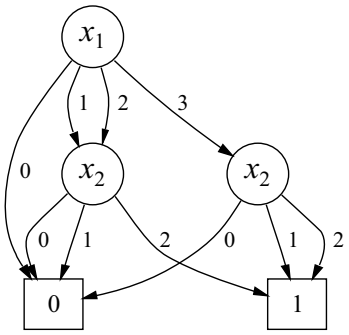

Figure 3: MDD example

node v whose successor nodes are all identical, and no two distinct nodes u, v exist such that the subgraphs rooted in u and v are isomorphic, i.e., for which: $var(u) = var(v)$, and $child_k(u) = child_k(v)$ for all $k \in range(var(u))$. For a given variable ordering, the ROMDD representation of any function is uniquely determined (up to isomorphism), as a result of which several properties (e.g., functional equivalence, constant functions, etc.) become easily testable. Conceptually, a reduced decision diagram can be interpreted as the result of the repeated application of two types of transformations on a decision tree or graph: one reduction rule is to bypass and delete redundant nodes (*elimination rule*), the other is to share isomorphic subgraphs (*merging rule*). In Figure 4, both rules are illustrated for a simple binary example. Note that, in practice, efficient implementations of diagram operations are used that directly produce a reduced form as the diagrams are being built. From here on, we will use the term 'MDD' to denote ROMDDs in particular.

Over the years, several BDD packages have been developed, which implement and provide interfaces for the manipulation of BDDs (in our experiments, we have applied David Long's package (Long, 2003), developed at Carnegie Mellon University. Most often, MDDs are implemented indirectly using these same packages, by binary encoding multi-valued variables (as explained in (Kam et al., 1998)). Direct MDD implementations have also been proposed, e.g. in (Miller and Drechsler, 1998).

4 EMPIRICAL EVALUATION

In previous research, we applied neural network rule extraction methods to extract a set of propositional if-then rules from a trained neural network (Baesens et al., 2003a; Baesens et al., 2003b). The experiments were conducted on two real-life credit-scoring data sets and the publicly available German credit data set.

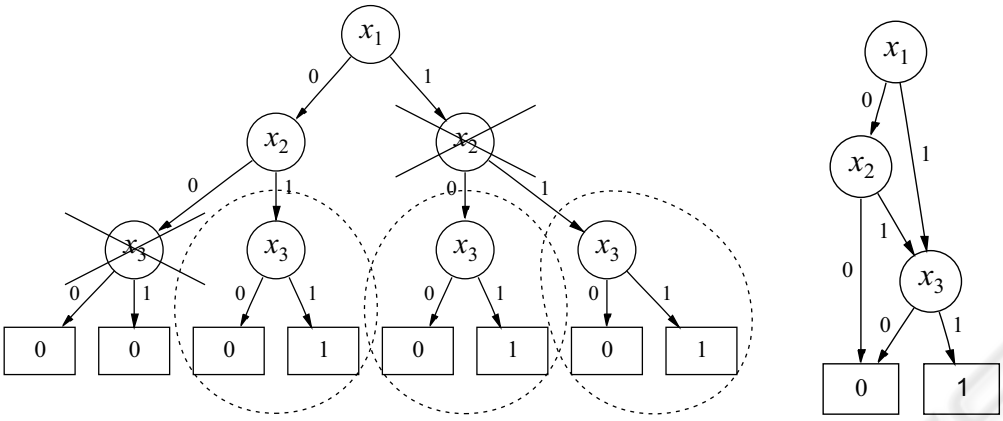

Figure 4: Decision trees (left) versus diagrams (right)

Figure 5 depicts the rule set that was extracted on the Bene1 data set obtained from a major Benelux financial institution (3123 Obs., 33 inputs).

If Term >12 months **and** Purpose = cash provisioning **and** Savings Account $\leq$ 12.40 € **and** Years Client $\leq$ 3 **then** Applicant = bad

If Term >12 months **and** Purpose = cash provisioning **and** Owns Property = no **and** Savings Account $\leq$ 12.40 € **then** Applicant = bad

If Purpose = cash provisioning **and** Income > 719 € **and** Owns Property = no **and** Savings Account $\leq$ 12.40 € **and** Years Client $\leq$ 3 **then** Applicant = bad

If Purpose = second-hand car **and** Income > 719 € **and** Owns Property = no **and** Savings Account $\leq$ 12.40 € **and** Years Client $\leq$ 3 **then** Applicant = bad

If Savings Account $\leq$ 12.40 € **and** Economical sector = Sector C **then** Applicant = bad

Default class: Applicant = good

Figure 5: Rules extracted for Bene1

It was shown that the extracted rule sets achieve a very high classification accuracy on independent test set data. The rule sets are both concise and easy to interpret and thus provide the credit-scoring expert with an insightful explanation. However, while propositional rules are an intuitive and well-known formalism to represent knowledge, they are not necessarily the most suitable representation in terms of structure and efficiency of use in case-by-case decision making. Research in knowledge representation suggests that graphical representation formalisms can be more readily interpreted and consulted by humans than a set of symbolic propositional if-then rules (see e.g. (Santos-Gomez and Darnel, 1992)).

Decision tables provide an alternative way of representing the extracted knowledge (Wets, 1998). We have used the PROLOGA (Prologa, 2003) software to construct the decision tables for the extracted rule sets. PROLOGA is an interactive modelling tool for computer-supported construction and manipulation of DTs (Vanthienen and Dries, 1994). A powerful rule language is available to help specify the DTs, and automated support is provided for several restructuring and optimization tasks.

Table 1 summarizes the properties of the DTs built from the extracted rule sets for the Bene1 and German credit data sets. For German credit (Bene1), the fully expanded decision table contained 6600 (192) columns, respectively. Subsequently, we converted each of these expanded DTs into a more compact DT, by joining nominal attribute values that do not appear in any rule antecedent into a common 'other' state, and then performing optimal table contraction (using a simple exhaustive search method). As a result of this reduction process, we ended up with two minimum-size contracted DTs, consisting of 11 and 14 columns for the German credit and Bene1 data sets, respectively (cf. right column of Table 1). Figure 6 depicts the resulting decision table for the Bene1 data set. While retaining the predictive accuracy of the original rule set, the top-down readability of such a DT, combined with its conciseness, makes the latter a very attractive visual representation of the extracted knowledge. Furthermore, the DT can be easily verified: clearly, there are no missing rules or inconsistencies in Figure 6.

However, a well-known property that can undermine the visual interpretability of decision trees, and hence also of lexicographically ordered DTs, is the inherent replication of subtrees or -tables implementing terms in disjunctive concepts (e.g. (Kohavi, 1996)).

Table 1: The number of columns in the expanded and minimized DTs.

Data set	Columns in expanded DT	Columns in minimized DT
German	6600	11
Bene1	192	14

Table 2: MDD size results

Data set	Intern. nodes in MDD (1)	Intern. nodes in dec. tree (2)	Size saving
German	8	8	0%
Bene1	9	12	25%

For example, in the DT for Bene1 (cf. Figure 6), column blocks $\{2, 3, 4, 5\}$ and $\{9, 10, 11, 12\}$, though having the same respective action values, are not eligible for contraction, because they differ in more than one condition entry (viz., with respect to the attributes 'purpose' and 'term'). On the other hand, a decision diagram, which allows the sharing of one such instance through multiple incoming edges, might be smaller than the corresponding tree or table. Therefore, in addition to the DT, we have built an equivalent MDD representation based on the extracted rule set, thereby adhering to the same ordering of attributes as in the minimum-size DT. Figure 7 presents the resulting diagram for Bene1. It was produced using the Graphviz graph-drawing software (Gansner et al., 1993; AT&T, 2003).

Unlike in Figure 6, the part of the MDD representation that matches the replicated table segment is included only once: the subgraph rooted at the rightmost of the two 'years client'-nodes is effectively shared through its two incoming edges. Hence, describing the function in MDD format results in a more compact representation, because the merging rule, unlike the DT contraction rule, does apply here. This empirically confirms why we consider a decision diagram to be a valuable alternative knowledge visualization aid. Nevertheless, decision diagrams are so far seldom considered in this context, despite their being a graph-based generalization of the far more frequently applied decision tree representation.

We have repeated the same exercise for the German credit data set, but in the latter case, no further size savings could be obtained vis-à-vis the DT representation. In Table 2, we have summarized the results of the MDD construction process for both data sets. Note that, because of the aforementioned relation between a decision tree and a lexicographically ordered DT, the figures in column (2) also match the number of splits appearing in the condition entry part of the corresponding DT. Consequently, the final column provides a measure of the additional size gains of MDD reduction over DT contraction (i.e., the added effect of graph sharing).

Both decision tables and diagrams facilitate the development of powerful decision-support systems that can be integrated in the credit-scoring process. A DT consultation engine typically traverses the table in a top-down manner, inquiring the user about the condition states of every relevant condition encountered along the way. A similar decision procedure is induced when consulting the decision diagram representation, and following the proper path through the graph. Hence, both types of representations provide efficient decision schemes that allow a system implementation to ask targeted questions and neglect irrelevant inputs during the question/answer-dialog. Furthermore, given the availability of efficient condition reordering operations for both types of representations, questions can be easily postponed during this process. For example, in PROLOGA, the available answer options always include an additional 'unknown' option, which allows the user to (temporarily) skip the question. When that happens, the DT's conditions are first reordered internally: moving the corresponding condition to the bottom of the order and then recontracting the DT may result in new don't care entries being formed for it. After that, the session continues with the next question. If, at some point, a conclusion is reached regarding the DT's actions, the former question could effectively be avoided; else, it eventually pops up again.

In the Bene1 example, suppose that we are deciding on a particular applicant whose properties will eventually be found to match against the condition entries of column 12, which tells us to accept the loan. Before arriving at that conclusion, we are required to provide only 4 of the 7 inputs to make a classification decision: 'term', 'owns property' and 'income' successively turn out to be irrelevant for this case. If, on the other hand, we would consult the rule description shown in Figure 5, we would need to evaluate every single rule, thereby testing its antecedent until a condition is found that fails, before we may conclude that none of the rules applies and that the default class (applicant = good) must be chosen.

5 CONCLUSIONS

In this paper, we have shown how credit-scoring knowledge can be compactly visualized either in the form of decision tables or diagrams. For two real-life cases, it was first of all shown how a set of propositional if-then rules, extracted by a neural network rule extraction algorithm, can be represented as a decision table. The constructed decision tables

1. Savings Account	$\leq$12.40 €													$>$ 12.40 €
2. Economical sector	Sector C	other												-
3. Purpose	-	cash provisioning							second-hand car				other	-
4. Term	-	$\leq$ 12 months				$>$ 12 months			-					
5. Years Client	-	$\leq$ 3			$>$3	$\leq$ 3		$>$3	$\leq$ 3			$>$ 3		
6. Owns Property	-	Yes	No		-	-	Yes	No	Yes	No		-	-	-
7. Income	-	-	$\leq$ 719 €	$>$ 719 €	-	-	-	-	-	$\leq$ 719 €	$>$ 719 €	-	-	-
1. Applicant=good	-	×	×	-	×	-	×	-	×	×	-	×	×	×
2. Applicant=bad	×	-	-	×	-	×	-	×	-	-	×	-	-	-
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 6: Decision table for the rules extracted for Bene1.

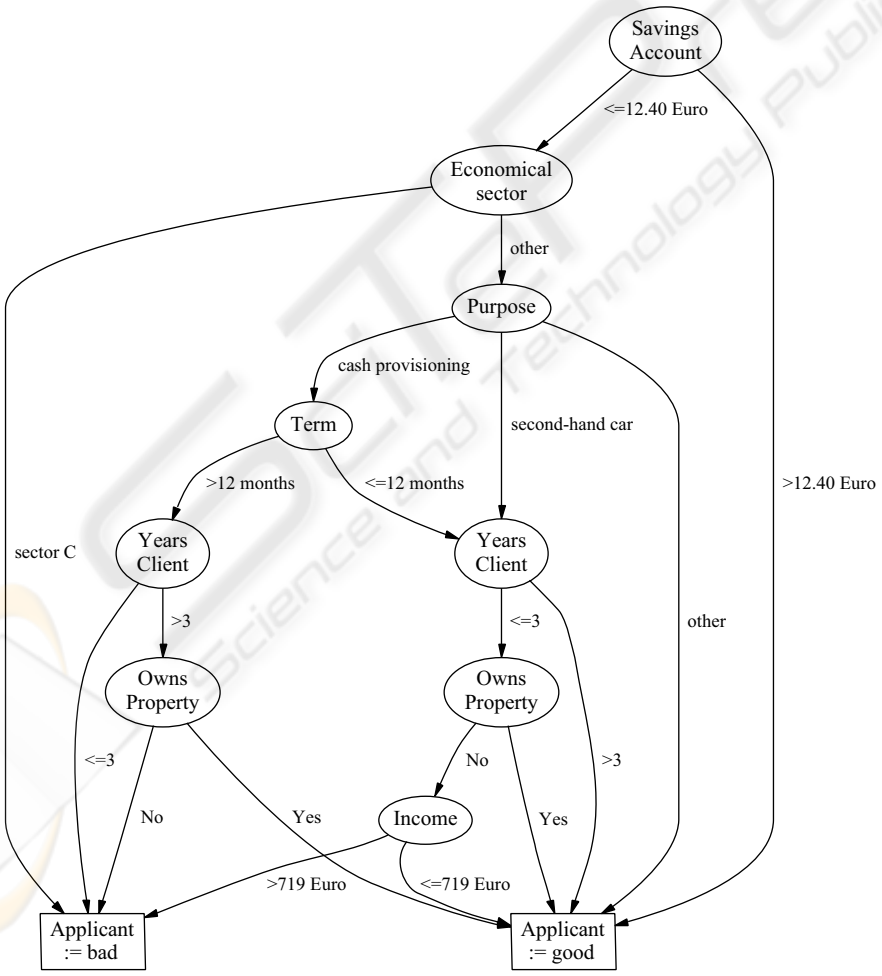

Figure 7: MDD for the Bene1 data set

were then reduced in size using lexicographical order-preserving contraction and condition row order minimization routines which in both cases yielded a parsimonious representation of the extracted rules, while preserving their predictive power. Secondly, we have advocated the use of multi-valued decision diagrams as an alternative visualization that can provide additional size savings compared to the former DT representation. What's more, we have seen empirical confirmation of this property on one of the two data sets. Subsequently, we have demonstrated that the use of either decision tables or diagrams facilitates an efficient case-by-case consultation of the knowledge (e.g., by limiting the number of questions that the user must answer in order to reach a conclusion). Hence, using decision tables and/or diagrams in combination with a rule extraction algorithm provides an interesting approach for developing powerful yet transparent credit-scoring models.

REFERENCES

AT&T Laboratories (2003). Graphviz, graph-drawing tool. http://www.research.att.com/sw/tools/graphviz/.

Baesens, B., Mues, C., Setiono, R., De Backer, M., and Vanthienen, J. (2003a). Building intelligent credit scoring systems using decision tables. In *Proceedings of the Fifth International Conference on Enterprise Information Systems (ICEIS'2003)*, pages 19–25, Angers, France.

Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. (2003b). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., and Vanthienen, J. (2003c). Benchmarking state of the art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635.

Bryant, R. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691.

David, R., Edelman, D., and Gammerman, A. (1992). Machine learning algorithms for credit-card applications. *IMA Journal of Mathematics Applied In Business and Industry*, 4:43–51.

Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K. P. (1993). A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230.

Henley, W. and Hand, D. (1997). Construction of a k-nearest neighbour credit-scoring system. *IMA Journal of Mathematics Applied In Business and Industry*, 8:305–321.

Horiyama, T. and Ibaraki, T. (2002). Ordered binary decision diagrams as knowledge-bases. *Artificial Intelligence*, 136(2):189–213.

Kam, T., Villa, T., Brayton, R. K., and Sangiovanni-Vincentelli, A. L. (1998). Multi-valued decision diagrams: Theory and applications. *International Journal on Multiple-Valued Logic*, 4(1-2):9–62.

Kohavi, R. (1996). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Department of Computer Science, Stanford University.

Long, D. (2003). OBDD package. http://www-2.cs.cmu.edu/∼modelcheck/bdd.html.

Miller, D. and Drechsler, R. (1998). Implementing a multi-valued decision diagram package. In *Proceedings of the International Symposium on Multiple-Valued Logic*, pages 52–57, Fukuoka, Japan.

Prologa (2003). Decision table modelling tool. http://www.econ.kuleuven.ac.be/prologa/.

Santos-Gomez, L. and Darnel, M. (1992). Empirical evaluation of decision tables for constructing and comprehending expert system rules. *Knowledge Acquisition*, 4:427–444.

Steenackers, A. and Goovaerts, M. (1989). A credit scoring model for personal loans. *Insurance: Mathematics and Economics*, 8:31–34.

Thomas, L. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to customers. *International Journal of Forecasting*, 16:149–172.

Vanthienen, J. and Dries, E. (1994). Illustration of a decision table tool for specifying and implementing knowledge based systems. *International Journal on Artificial Intelligence Tools*, 3(2):267–288.

Vanthienen, J., Mues, C., and Aerts, A. (1998). An illustration of verification and validation in the modelling phase of kbs development. *Data and Knowledge Engineering*, 27:337–352.

Wets, G. (1998). *Decision Tables in Knowledge-Based Systems: Adding Knowledge Discovery and Fuzzy Concepts to the Decision Table Formalism*. PhD thesis, Department of Applied Economic Sciences, Catholic University of Leuven, Belgium.