DEPARTMENT OF APPLIED ECONOMICS

RESEARCH REPORT

# DECISION DIAGRAMS IN MACHINE LEARNING: AN EMPIRICAL STUDY ON REAL-LIFE CREDIT-RISK DATA

CHRISTOPHE MUES • BART BAESENS • CRAIG M. FILES • JAN VANTHIENEN

OR 0405

# Decision Diagrams in Machine Learning: an Empirical Study on Real-Life Credit-Risk Data

Christophe Mues[1], Bart Baesens[1,2], Craig M. Files[3], Jan Vanthienen[1]

[1]K.U.Leuven, Dept. of Applied Economic Sciences,

Naamsestraat 69, B-3000 Leuven, Belgium

{Christophe.Mues; Bart.Baesens; Jan.Vanthienen}@econ.kuleuven.ac.be

[2]University of Southampton, School of Management

Southampton, SO17 1BJ, United Kingdom

b.m.m.baesens@soton.ac.uk

[3]3100 San Luis St., Fort Collins,

Colorado 80525-6612

cfiles@ee.pdx.edu

**Abstract**

Decision trees are a widely used knowledge representation in machine learning. However, one of their main drawbacks is the inherent replication of isomorphic subtrees, as a result of which the produced classifiers might become too large to be comprehensible by the human experts that have to validate them. Alternatively, decision diagrams, a generalization of decision trees taking on the form of a rooted, acyclic digraph instead of a tree, have occasionally been suggested as a potentially more compact representation. Their application in machine learning has nonetheless been criticized, because the theoretical size advantages of subgraph sharing did not always directly materialize in the relatively scarce reported experiments on real-world data. Therefore, in this paper, starting from a series of rule sets extracted from three real-life credit-scoring data sets, we will empirically assess to what extent decision diagrams are able to provide a compact visual description. Furthermore, we will investigate the practical impact of finding a good attribute ordering on the achieved size savings.

# 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 these data and build models that distinguish good payers from bad payers using characteristics such as amount on savings account, marital status, purpose of loan, etc. Many classification techniques have been suggested in the literature to build credit-scoring models [2, 22]. Amongst the most popular are traditional statistical methods (e.g. logistic regression [21]), nonparametric statistical models (e.g. k-nearest neighbor [13] and classification trees [5]) and neural networks [6]. Especially neural networks have in recent years received a lot of attention. However, a major drawback is the lack of transparency of the resulting models. While they are generally able to achieve a high predictive accuracy rate, the reasoning behind how they reach their decisions is not readily available, which hinders their acceptance by practitioners. As a result, one often sees that the estimated credit-scoring models fail to be successfully integrated into the actual decision environment.

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, such as decision tables or trees [1]. The latter notations are intended to communicate the extracted knowledge to the credit-scoring expert in a format that he/she can more easily understand and validate, and efficiently apply in every-day practise. In our experience, the ability to provide such a visualization has become a critical success factor for the development of decision-support systems for credit scoring.

Clearly, an important criterion where human interpretability is concerned, is the size of the generated representation. Despite their being intuitive and efficiently applicable in theory, it has regularly been observed that the decision trees generated by machine-learning algorithms turn out to be too large to be comprehensible to human experts.

In that regard, one of their main limitations is the inherent replication of isomorphic subtrees implementing terms in disjunctive concepts. Hence, in this paper, we report on the alternative use of *decision diagrams*. The latter are a generalization of decision trees taking on the form of a rooted, acyclic digraph instead of a tree, which have to a great extent been studied and applied by the hardware design community [3]. Their use has also occasionally been proposed in the machine-learning community (e.g. in [16, 17, 18]), precisely because of the potential size savings a graph-based representation might offer over a tree-based one.

Nevertheless, decision diagrams have so far not gained wide acceptance in the latter problem context, partly because the theoretical size advantages of subgraph sharing did not always directly materialize in (the relatively scarce) reported experiments [7]. Two problems that seem to have impaired a thorough empirical study are: (1) the use of very different learning algorithms for the respective types of representations being compared (thus making it hard to separate the effect of the representation from that of the specific algorithm); (2) the impact of attribute ordering on the size of the resulting description. Therefore, in this paper, starting from a series of rule sets produced from real-life credit-scoring data by neural network rule extraction, we will empirically assess to what extent decision diagrams are able to provide a more compact visual description than their decision tree counterparts. Furthermore, we will investigate the practical impact of finding a good attribute ordering on the achieved size savings.

This paper is organized as follows. Section 2 discusses the basic concepts of decision diagrams and how they may provide an alternative, more concise view of the extracted knowledge. The empirical setup and results are presented in section 3. Finally, section 4 concludes the paper.

# 2 Decision Diagrams

Decision diagrams are a graph-based representation of discrete functions, accompanied by a set of graph algorithms that implement operations on these functions. 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 [3], 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 [14] and machine learning [9, 16, 17, 18], while their use for the visual representation of rules extracted from neural networks, or in the particular research domain of credit scoring, has to our knowledge not been proposed before.

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 [15]. An MDD is a rooted, directed acyclic graph, with a sink node for every possible output value (class). Each internal node $v$ is labelled by a test variable (attribute) $var(v) = x_i \, (i = 1, ..., n)$, 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 $x_1 \prec x_2 \prec ... \prec x_n$; i.e., for each edge leading from a node labelled by $x_i$ to a node labelled by $x_j$, it holds that $x_i \prec x_j$.

An OMDD is meant to represent an $n$-variable discrete (classification) 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 case. Figure 1 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 $x_1 \prec x_2$.

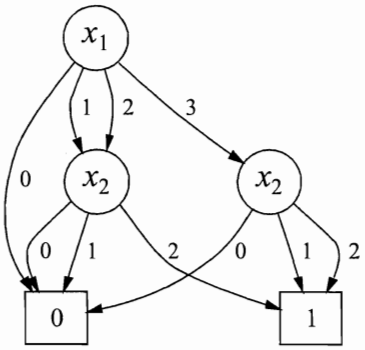



Figure 1: MDD example

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 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 2, 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' or decision diagram to denote

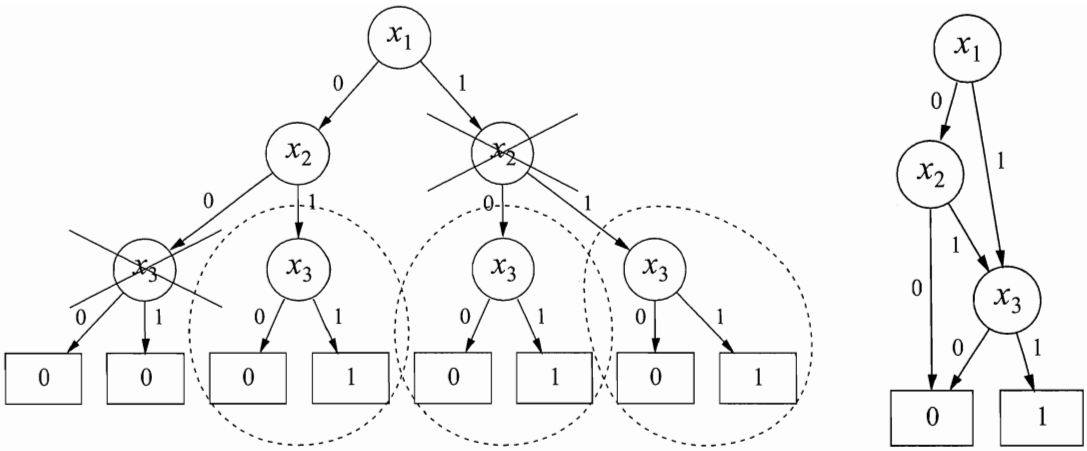



Figure 2: Decision trees (left) versus diagrams (right)

ROMDDs in particular.

When using decision diagrams to represent a function, some total ordering of the input variables must be selected. Since the size of the resulting diagram (i.e., the number of nodes) is very sensitive to this choice, finding a suitable ordering is critical in many application domains. Figure 3 illustrates the extent to which form and size can be affected by the chosen variable order. Both diagrams shown represent the same function (given by the Boolean formula $x_{1,1}x_{1,2} + x_{2,1}x_{2,2} + x_{3,1}x_{3,2}$), but using different orders. Several exact minimization algorithms have been proposed (e.g. in [10]), but, considering that finding an optimal order is an NP-hard problem, they are often too costly for larger problem instances (i.e., with many variables). Hence, heuristic approaches (selecting some ordering based on available problem data) or local search techniques (which aim at improving a given variable order, e.g., by moving variables up or down the graph) have been widely investigated as well.

Over the years, several BDD packages have been developed, which implement and provide interfaces for the manipulation of BDDs. Most often, MDDs are implemented indirectly using these same packages, by binary encoding multi-valued variables. Direct MDD implementations have also been proposed, e.g. in [9]. The latter package was used in the subsequent experiments.

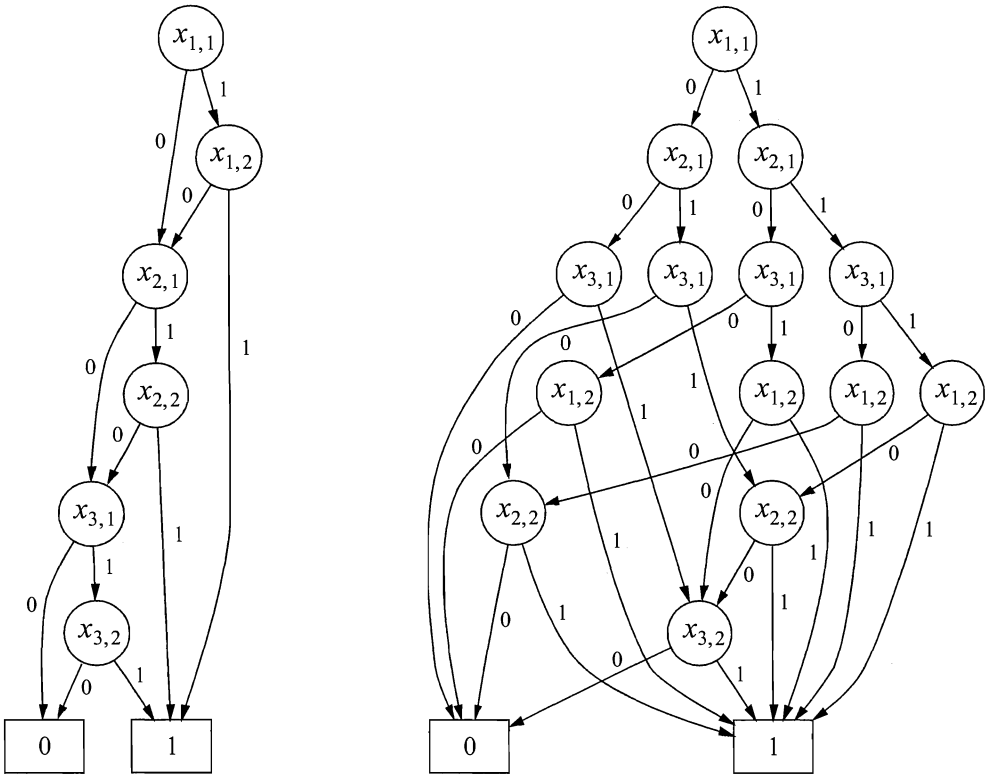



Figure 3: Effect of variable ordering on decision diagram size; example taken from [3]

# 3 Empirical Evaluation on Real-Life Credit-Scoring Data

## 3.1 Step 1: Neural Network Rule Extraction

The experiments were conducted on three real-life credit-risk evaluation data sets: German credit, Bene1 and Bene2. The Bene1 and Bene2 data sets were obtained from two major Benelux (Belgium, The Netherlands, Luxembourg) financial institutions. The German credit data set is publicly available at the UCI repository (`http://www.ics.uci.edu/~mlearn/MLRepository.html`). All data sets were discretized using the discretization algorithm of Fayyad and Irani with the default options [8].

We then investigated the performance of two neural network rule (tree) extraction algorithms: *Neurorule* and *Trepan*. Neurorule starts by training and pruning a neural network for the given classification task. It then extracts a set of propositional if-then rules, which mimics the decision process of the neural network and resolves its black box property (see [1, 20] for more details). For example, Figure 4 displays the rules extracted by Neurorule on the Bene1 data set. Trepan is a neural network tree extraction algorithm which tries to approximate the neural network as a decision tree whose nodes may consist of $m$-of-$n$ expressions [4]. The tree is grown recursively using information-theoretic concepts. The neural network is hereby used as an oracle to generate additional observations, when the number of data points available to decide upon the splits becomes unacceptably low.

The performance of both neural network extraction algorithms was contrasted with that of the neural network itself and with three other algorithms producing decision trees, rules and diagrams. C4.5 is a well-known induction algorithm which uses information-theoretic concepts to grow a decision tree [19]. It first grows a full tree and then retrospectively prunes it in order to avoid overfitting. The C4.5rules algorithm converts this tree to a set of rules which can then be further pruned [19]. The EODG (Entropy-based

**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 4: Rules for Bene1 extracted by Neurorule

Oblivious Decision Graphs) algorithm uses mutual information to build a decision tree in a top-down manner; this tree is subsequently converted to a decision diagram by merging its isomorphic subtrees [17].

Table 1 presents the classification performance of the discussed techniques on the three credit-scoring data sets. Note that the reported accuracy was computed on independent test sets (typically one-third of the observations) and thus adequately represents the generalization behavior of the classification technique. It can be observed that the

| Data set | Method | Accuracy | Complexity |
|---|---|---|---|
| **German** | Neural network | 77.84 | 6 inputs |
| | Neurorule | 77.25 | 4 rules |
| | Trepan | 73.95 | 21 nodes |
| | C4.5 | 71.56 | 54 nodes |
| | C4.5rules | 74.25 | 17 rules |
| | EODG | 72.45 | 9 nodes |
| **Bene1** | Neural network | 71.85 | 7 inputs |
| | Neurorule | 71.85 | 6 rules |
| | Trepan | 71.85 | 21 nodes |
| | C4.5 | 70.03 | 114 nodes |
| | C4.5rules | 70.12 | 17 rules |
| | EODG | 71.37 | 5 nodes |
| **Bene2** | Neural network | 74.09 | 7 inputs |
| | Neurorule | 74.13 | 7 rules |
| | Trepan | 74.01 | 17 nodes |
| | C4.5 | 73.09 | 578 nodes |
| | C4.5rules | 73.51 | 27 rules |
| | EODG | 72.38 | 7 nodes |

Table 1: Classification accuracy of rule, tree and diagram extraction techniques

Neurorule and Trepan algorithms fairly well approximate the performance of the neural networks from which they were derived. For the Bene2 data set, the Neurorule method even outperforms the neural network slightly. Both algorithms consistently yield very good classification performance when compared to C4.5, C4.5rules and EODG. Besides classification performance, we also report the number of inputs, extracted rules or nodes.

When looking at these criteria, it becomes clear that Neurorule and Trepan extract concise decision models. Although the EODG algorithm extracts very concise representations as well, its classification performance is inferior when compared to Neurorule and Trepan. The size of the C4.5-tree is in all cases prohibitively large for visualization purposes.

Although the knowledge descriptions extracted by Neurorule or Trepan already offer an insightful explanation of the neural network model they were generated from, they lack an efficient evaluation scheme by which the expert can validate the knowledge as a whole, or apply it to case-by-case decision making. For those purposes, diagrammatic notations such as decision tables, trees or diagrams, instead of being induced directly, can additionally provide a more suited visualization of the extracted rule sets. Thus, the format in which the knowledge is being communicated can be transformed without causing any loss of predictive accuracy. This idea will be elaborated on next.

## 3.2 Step 2: Visualizing the Extracted Knowledge using Decision Diagrams

In previous work [1], we have largely focused on the use of a particular class of (lexicographically ordered) decision tables in this subsequent knowledge visualization step. It was shown that this restricted type of decision table exhibits very similar properties to a decision tree, in that it can be efficiently evaluated in a top-down manner. Rather than having to evaluate the textual rule expressions one by one, the credit-scoring expert can thus quickly reach a conclusion for a given application by following the proper path through this tree structure. For example, in Figure 5, a decision tree is shown that is functionally equivalent to the prior rule set of Figure 4 (i.e., it classifies all possible applicants in the same way).

While retaining the predictive accuracy of the original rule set, the top-down readability of a decision tree makes it a seemingly attractive visual representation of the extracted knowledge. However, a well-known property that can undermine the conciseness and interpretability of decision trees (and hence also of lexicographically ordered

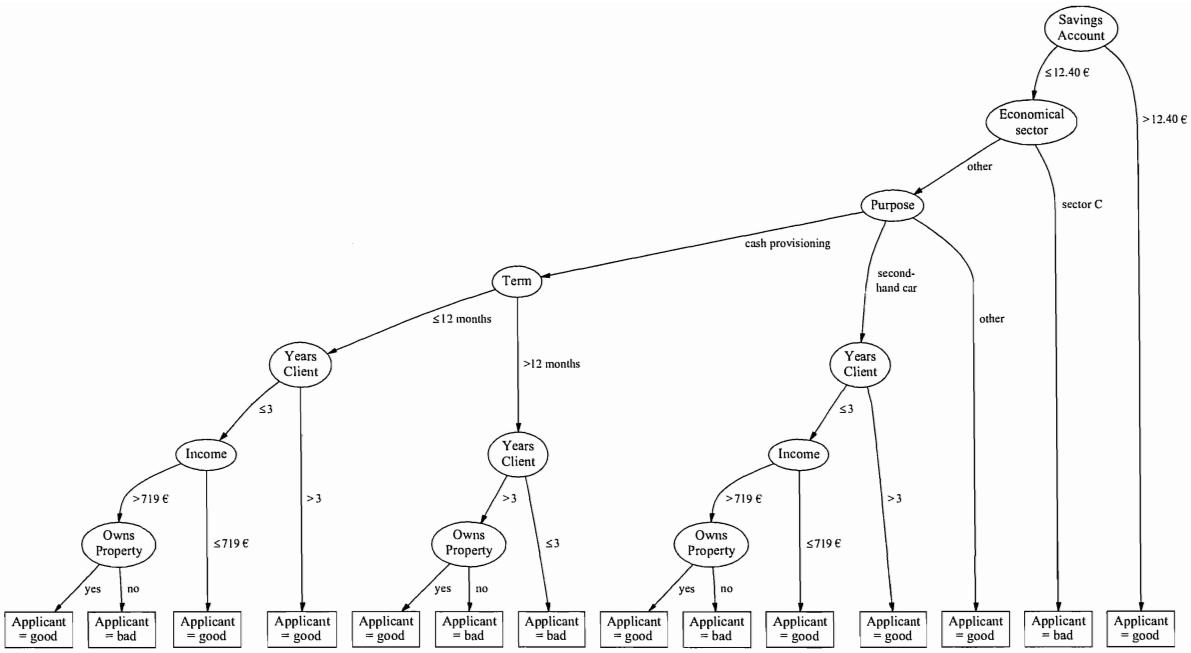



Figure 5: Example of an (ordered) decision tree for Bene1/Neurorule

decision tables) is the inherent replication of subtrees implementing terms in disjunctive concepts (as explained, e.g., in [16]). This is the reason why we have decided to also investigate decision diagrams as an alternative representation that could help avoid such unnecessary replication, provided that a suitable attribute ordering can be found for the problem at hand. The principal goal of this study therefore is to empirically verify these theorized advantages in a real-life credit-scoring setting.

For example, although the tree shown in Figure 5 is substantially smaller than the corresponding C4.5-tree (cf. Table 1), it still contains a certain degree of replication. Most notably, two out of the three subtrees rooted at a 'years client'-test node are isomorphic. In contrast, by reducing the tree into a decision diagram, in which recurring parts are shared through multiple incoming edges, a smaller representation is obtained (cf. Figure 6, below).[1] In the latter, the subgraph rooted at the rightmost of the two

[1]To produce these graph drawings, we used the Graphviz software [12] from AT&T Laboratories (`http://www.research.att.com/sw/tools/graphviz`).

'years client'-nodes is thus included once instead of twice. If these size savings are indeed substantial on average, a decision diagram will provide a valuable alternative knowledge visualization.

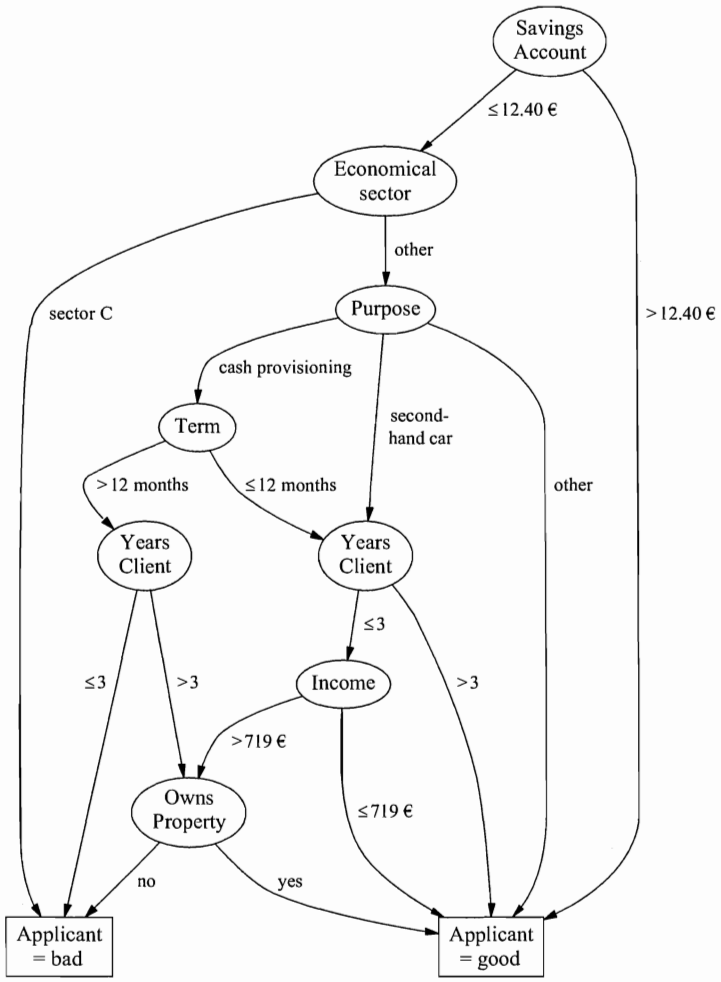



Figure 6: Minimum-size MDD for Bene1/Neurorule

Hence, we further processed each rule set by joining nominal attribute values that do not appear in any rule antecedent into a common 'other' state, and by rewriting rules containing negations or $m$-of-$n$ expressions into disjunctive normal form. Based on the latter format, we then built a decision diagram representation, using the standard implementations of logical sum and product provided by the MDD-package. As explained

in section 2, the size of the resulting diagram depends on the order in which the attributes are evaluated. To find an optimal order (i.e., which results in a minimum-size MDD), we implemented a simple exhaustive search procedure, at every step of which two neighboring variables in the order are swapped. Considering that adjacent variable pairs can be swapped efficiently by a local exchange of subgraphs [11], and given the input space reduction achieved in the preceding step of the knowledge discovery process, execution turned out to be feasible (other more efficient minimization algorithms are described elsewhere, e.g. in [10]). As a result of this optimization process, we ended up with a minimum-size MDD for each rule set. Figure 6 earlier depicted the MDD thus obtained from the Bene1 rule set extracted by Neurorule. The results for all MDDs are listed in Table 2. Most importantly, in all cases, the diagrams were sufficiently concise to be easily understood and applied.

| **Data set** | **Extraction method** | **Internal nodes in min.-size MDD** | **Internal nodes in matching tree** | **Size saving** |
|---|---|---|---|---|
| German | Neurorule | 7 | 14 | 50% |
| | Trepan | 7 | 7 | 0% |
| Bene1 | Neurorule | 8 | 12 | 33.3% |
| | Trepan | 14 | 29 | 51.7% |
| Bene2 | Neurorule | 11 | 28 | 60.7% |
| | Trepan | 16 | 51 | 68.6% |

Table 2: MDD size results

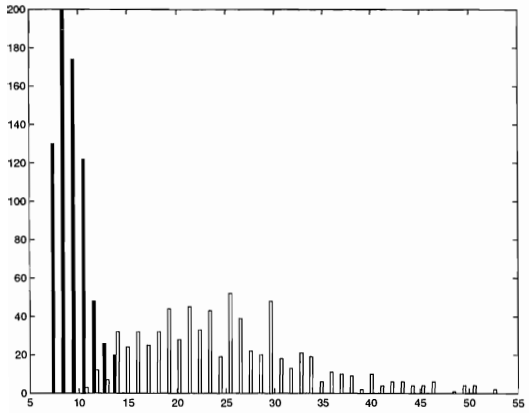
In Table 2, we can also see that, except for the German credit classifier produced by Trepan, substantial size gains are being achieved as a result of MDD reduction (unlike, e.g., for the learning algorithm applied in [7], which reportedly produced few node merging on real-world data sets). To give an idea of the amount of subgraph sharing, we have included a column displaying the size of the equivalent decision tree obtained when the same (total) attribute ordering is adopted (note that we are not considering unordered trees or graphs at this point). To make the analysis fair, we avoid repetitive counting of sink nodes, and measure size in terms of the number of internal nodes. The

percentage in the final column thus provides an indication of the effectiveness of the merging rule.

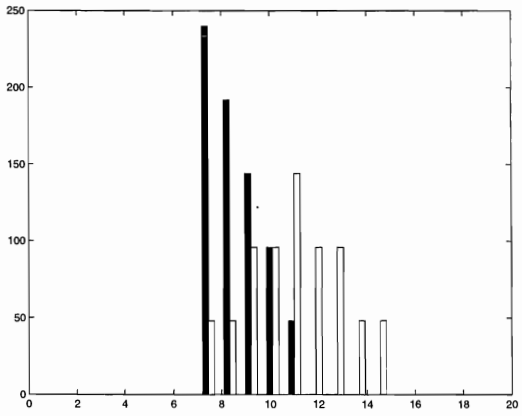
As explained above, the reported figures are for minimum-size MDDs. Unlike in prior decision diagram based learning approaches, we do not have to revert to a greedy ordering strategy, or to incremental reordering methods (as in [18]), because MDDs are applied only after the input space has been drastically reduced in the first step of the process. Consequently, we are able to more fully explore the impact of variable ordering on diagram size. Figure 7 displays the observed size distribution for all investigated cases. Along the y-axis of each bar plot, the number of condition orders is indicated that lead to the number of internal nodes specified on the x-axis. The resulting distribution curve for the MDDs is depicted by solid boxes; empty boxes indicate the same relation for the matching decision trees (i.e., without subgraph merging). Although, even with non-optimal attribute orders, the MDDs obtained still are relatively small on average (most points on the MDD curves are well to the left of the tree size curve), the importance of finding an appropriate attribute ordering becomes clear. For example, for Bene1/Trepan, the number of internal nodes varies from 14 to 38. Obviously, where comprehensibility and evaluation efficiency are concerned, this is an important difference. Hence, as could be expected, the utility of decision diagram techniques as a visual communication aid strongly depends on whether an adequate ordering strategy is applied.

# 4 Conclusions and Future Work

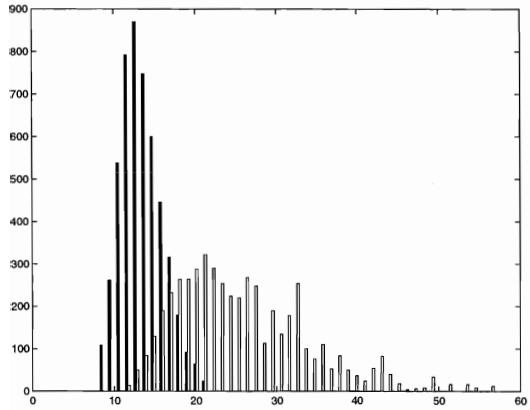
In this paper, we have demonstrated the effectiveness of a two-step approach to build accurate yet comprehensible credit-scoring models from data, using three real-life data sets. Firstly, powerful rule set classifiers were obtained using neural network rule extraction techniques (viz., Neurorule and Trepan). In the second step, these were then compactly visualized in the form of decision diagrams, thus providing the credit-scoring expert with a comprehensible and efficiently applicable notation, while retaining the



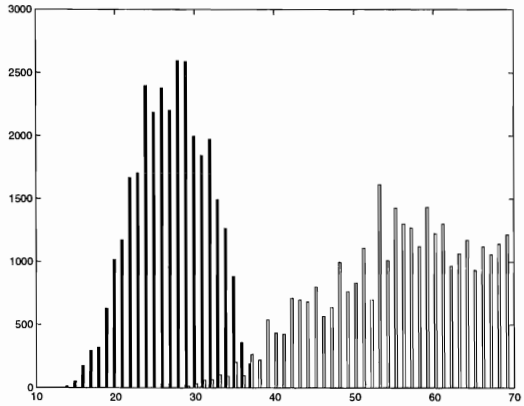
(a) German/Neurorule



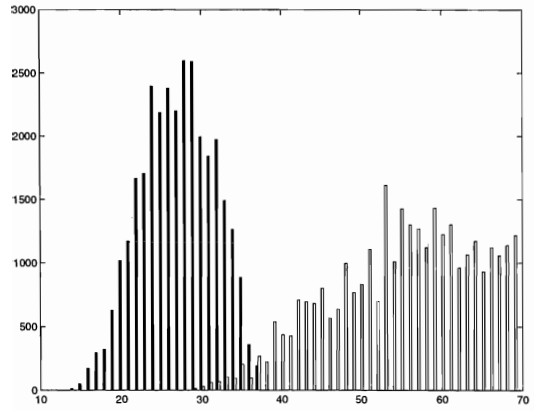
(b) German/Trepan



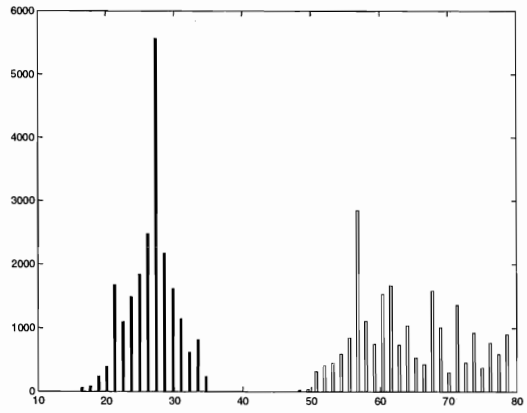
(c) Bene1/Neurorule



(d) Bene1/Trepan



(e) Bene2/Neurorule



(f) Bene2/Trepan

Figure 7: Size distribution of decision diagram vs. tree

predictive accuracy of the original rule set. To minimize the size of the resulting diagrams, an exact variable order optimization procedure was applied. In all cases, this approach yielded highly accurate classifiers, compared to the decision tree and diagram inducers C4.5 and EODG, while the resulting decision diagrams were also satisfactorily concise. We found that the MDD reduction mechanism was quite effective, in that several isomorphic subgraphs are being shared which would otherwise be replicated when using a decision tree representation. This is a noteworthy result, considering that the theorized advantages of decision diagram inducers did thus far not often directly materialize in a real-life setting. Finally, the importance of selecting a good attribute order was demonstrated.

Although the obtained decision diagrams are clearly more compact than their decision tree counterparts, it has to be noted that we have restricted the comparison to both ordered diagrams and trees (i.e., in which the order of testing variables does not differ between different branches). Clearly, the search space of finding a minimal unordered representation (sometimes also referred to as 'branching programs' or 'free' diagrams) is much larger, while the relative advantages of graph sharing might be less prominent.

Secondly, we have implicitly assumed that a compact graph representation is to be preferred over a larger tree-based representation. Obviously, overly large decision trees inhibit the intuitiveness and usability of the extracted knowledge, hence succinctness is indeed an important factor. However, an interesting topic for further research would be to investigate, in an experimental setting, to what extent credit-scoring experts are actually at ease interpreting these graph-based decision schemes as opposed to the more conventional tree-based schemes. In that case, additional evaluation criteria would, e.g., have to be the average time required by practitioners to classify applicants, or the observed frequency of classification mistakes.

# References


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

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

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

[4] M.W. Craven and J.W. Shavlik. Extracting tree-structured representations of trained networks. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 24–30, Cambridge, MA, U.S., 1996. MIT Press.

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

[6] V.S. Desai, J.N. Crook, and G.A. Overstreet Jr. A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research*, 95(1):24–37, 1996.

[7] T. Elomaa and M. Kääriäinen. On the practice of branching program boosting. *Lecture Notes in Artificial Intelligence*, 2167:133–144, 2001.

[8] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1022–1029, Chambéry, France, 1993. Morgan Kaufmann.

[9] C. Files. *A New Functional Decomposition Method As Applied to Machine Learning and VLSI Layout.* PhD thesis, Department of Electrical and Computer Engineering, Portland State University, 2000.

[10] S.J. Friedman and K.J. Supowit. Finding the optimal variable ordering for binary decision diagrams. *IEEE Transactions on Computers*, 39(5):710–713, 1990.

[11] M. Fujita, Y. Matsunaga, and T. Kakuda. On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In *Proceedings of the European Design Automation Conference*, pages 50–54, 1991.

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

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

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

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

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

[17] R. Kohavi and C.H. Li. Oblivious decision trees, graphs, and top-down pruning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1071–1077, Montréal, Québec, Canada, 1995. Morgan Kaufmann Publishers.

[18] A.L. Oliveira and A.L. Sangiovanni-Vincentelli. Using the minimum description length principle to infer reduced ordered decision graphs. *Machine Learning*, 25(1):23–50, 1996.

[19] J.R. Quinlan. *C4.5 programs for machine learning.* Morgan Kaufmann, 1993.

[20] R. Setiono and H. Liu. Symbolic representation of neural networks. *IEEE Computer*, 29(3):71–77, 1996.

[21] A. Steenackers and M.J. Goovaerts. A credit scoring model for personal loans. *Insurance: Mathematics and Economics*, 8:31–34, 1989.

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