Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification

Abraham Bernstein, Foster Provost, and Shawndra Hill

Abstract—A data mining (DM) process involves multiple stages. A simple, but typical, process might include preprocessing data, applying a data mining algorithm, and postprocessing the mining results. There are many possible choices for each stage, and only some combinations are valid. Because of the large space and nontrivial interactions, both novices and data mining specialists need assistance in composing and selecting DM processes. Extending notions developed for statistical expert systems we present a prototype Intelligent Discovery Assistant (IDA), which provides users with 1) systematic enumerations of valid DM processes, in order that important, potentially fruitful options are not overlooked, and 2) effective rankings of these valid processes by different criteria, to facilitate the choice of DM processes to execute. We use the prototype to show that an IDA can indeed provide useful enumerations and effective rankings in the context of simple classification processes. We discuss how an IDA could be an important tool for knowledge sharing among a team of data miners. Finally, we illustrate the claims with a demonstration of cost-sensitive classification using a more complicated process and data from the 1998 KDDCUP competition.

Index Terms—Cost-sensitive learning, data mining, data mining process, intelligent assistants, knowledge discovery, knowledge discovery process, machine learning, metalearning.

1 Introduction

Knowledge discovery from data is the result of an exploratory process involving the application of various algorithmic procedures for manipulating data, building models from data, and manipulating the models. The Knowledge Discovery (KD) process [1] is one of the central notions of the field of Knowledge Discovery and Data mining (KDD). The KD process deserves more attention from the research community: processes comprise multiple algorithmic components, which interact in nontrivial ways. Even data mining specialists are not familiar with the full range of components, let alone the vast design space of possible processes. Therefore, both novices and data mining specialists are apt to overlook useful instances of the KD process. We consider tools that will help data miners to navigate the space of KD processes systematically, and more effectively. In particular, this paper focuses on a subset of stages of the KD process—those stages for which there are multiple algorithm components that can apply; we will call this a data mining (DM) process (to distinguish it from the larger knowledge discovery process). For most of this paper, we consider a prototypical DM process template, similar to the one described by Fayyad et al. [1] and Chapman et al. [2], which is shown in Fig. 1. We concentrate our work here on three DM-process stages: automated preprocessing of data, application of induction algorithms, and automated postprocessing of models. We have chosen this set of steps because, individually, they are relatively well understood—and they can be applied to a wide variety of benchmark data sets.[1] In the final case study, we expand our view to a more involved DM process.

Fig. 2 shows three simple, example DM processes. Process 1 comprises simply the application of a decision-tree inducer. Process 2 preprocesses the data by discretizing numeric attributes, and then builds a naïve Bayesian classifier. Process 3 preprocesses the data first by taking a random subsample, then applies discretization, and then builds a naïve Bayesian classifier. Descriptions of all of the techniques can be found in a data mining textbook [5].

Consider an intelligent assistant (an *intelligent discovery assistant*, or IDA) that helps a data miner with the exploration of the space of valid DM processes. A *valid* DM process violates no fundamental constraints of its constituent techniques. For example, consider an implementation of a naïve Bayesian classifier that applies only to categorical attributes.[2] If an input data set contains numeric attributes, simply applying this classifier is not a valid DM process. However, Process 2 is valid, because it preprocesses the data with a discretization routine, transforming the numeric attributes to categorical ones.

1. More generally, because we will assemble these components automatically into complete processes that can be executed by a user, the scope of our investigation is necessarily limited to KD-process stages for which there exist automated components, and for which their requirements and functions can be specified. Important but ill-understood stages, such as "business process analysis" or "management of discovered knowledge," are not included [3]. We also do not consider intelligent support for more open-ended, statistical/exploratory data analysis, as has been addressed by St. Amant and Cohen [4].

2. The naïve Bayesian approach generally allows induction for data containing continuous attributes. However, treatment varies by implementation. For this paper (partially for demonstration), we will assume an implementation of naïve Bayes that does not aloow continuous variable without preprocessing.

- *A. Bernstein is with the Department of Informatics, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland. E-mail: bernstein@ifi.unizh.ch.*
- *F. Provost and S. Hill are with the Stern School of Business, New York University, 44 West 4th Street, New York, NY 10012. E-mail: {fprovost, shill}@stern.nyu.edu.*

Manuscript received 15 Apr. 2002; revised 15 July 2004; accepted 31 Aug. 2004; published online 17 Feb. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 116351.

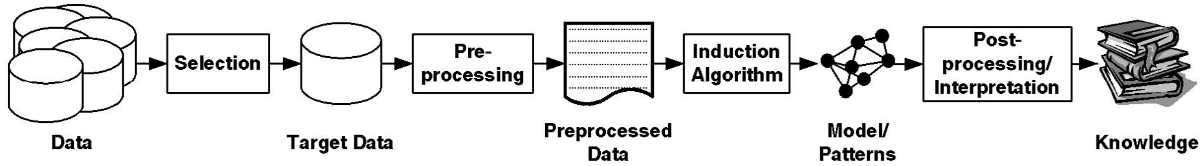

Fig. 1. The KD process (adapted from Fayad et al. [1]).

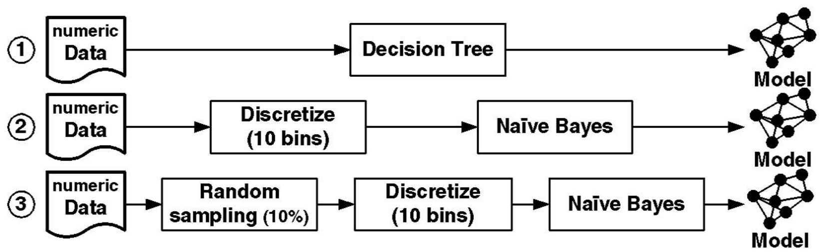

Fig. 2. Three valid DM processes.

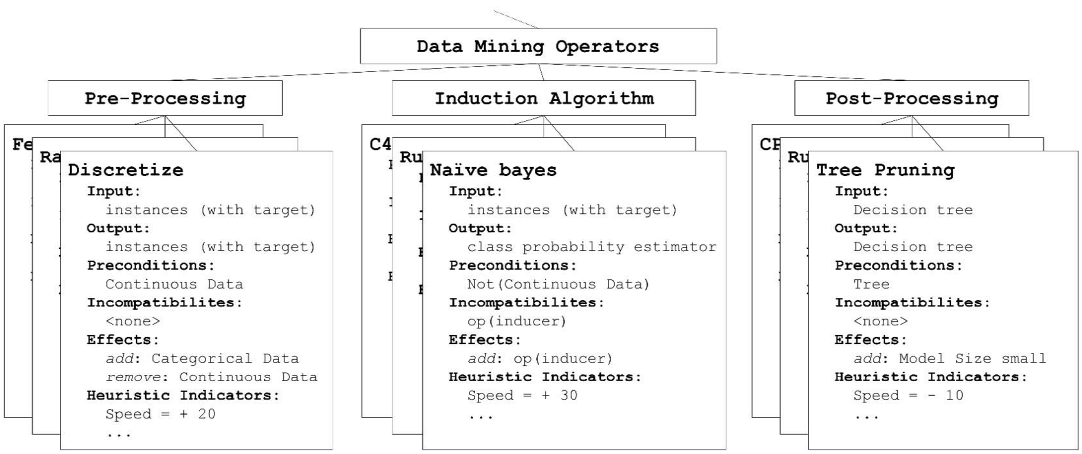

Fig. 3. Simplified elements of a DM ontology.

An automated system can take advantage of an explicit ontology of data mining techniques, which defines the various techniques and their properties. The IDA determines characteristics of the data and of the desired mining result, and uses the ontology to search for and enumerate the DM processes that are valid for producing the desired result from the given data. Each search operator corresponds to the inclusion in the DM process of a different data mining technique; preconditions constrain its applicability and there are effects of applying it. Fig. 3 shows some simplified ontology entries. The IDA also assists the user in choosing processes to execute, for example, by ranking the processes (heuristically) according to what is important to the user. The ranking shown in Fig. 2 (based on the number of techniques that form the plan) would be useful if the user were interested in minimizing fuss. Another user may want to minimize runtime. In that case, the reverse of the ranking shown in Fig. 2 would be better. Other ranking criteria are accuracy, cost sensitivity, comprehensibility, etc., and combinations thereof.

We claim that such a system can provide users with two main benefits:

1. a systematic enumeration of valid DM processes, so users do not miss important, potentially fruitful options, and
2. effective rankings of these valid processes by different criteria, to help users choose between the options.

We also assert that an ontology-based IDA provides an infrastructure for sharing knowledge about data mining processes, which can lead to what economists call network externalities. We do not provide experimental support for this third hypothesis, but argue that behavioral research in the area of knowledge sharing has shown such effects in analogous applications.

We support the first claim by presenting in detail the design of an effective IDA for cost-sensitive classification, including a working prototype, describing how valid plans are enumerated based on an ontology that specifies the characteristics of the various component techniques. We show plans that the prototype produces, and argue that they would be useful not only to novices, but even to expert data miners. We provide support for the second claim with an experimental study, using ranking heuristics. Although we do not claim to give an in-depth treatment of ranking methods, we demonstrate the ability of the IDA prototype to rank potential processes by speed and by accuracy (both of which can be assessed objectively) and by combinations of the two, in the context of a classification task. Finally, we provide additional support for all the claims with an empirical demonstration, using the KDDCUP 1998 data

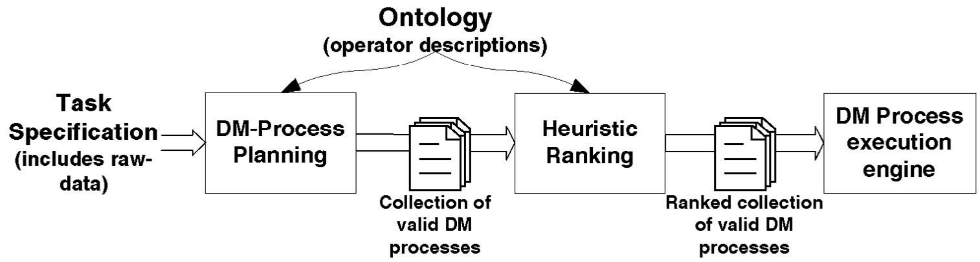

Fig. 4. The overall process followed by an IDA.

mining problem, showing how an IDA can take advantage of knowledge about a problem-specific DM process.

2 Motivation and General Procedure

When engaged in design activities, people rarely explore the entire design space [6]. When confronted with a new problem, data miners, even data mining experts, often do not explore the design space of DM processes thoroughly. For example, the ACM SIGKDD Conference holds an annual competition, in which a never-before-seen data set is released to the community and teams of researchers and practitioners compete to discover knowledge from the data. KDDCUP-2000 received 30 entrants (teams) attempting to mine knowledge from electronic-commerce data. As reported by Kohavi et al. [7], most types of data mining algorithm were tried by only a small fraction of participants.

Expert data miners may ignore many data mining approaches because they do not have access to the tools; however, readily and freely available data mining toolkits make this reason suspect. More likely, experts simply do not use many data mining tools—especially tools that require additional pre and postprocessing or those requiring complicated procedures for installation or execution (e.g., complicated parameter tweaking). Indeed, the only algorithm that was tried by more than 20 percent of the KDDCUP-2000 participants was decision-tree induction, which often performs reasonably well with no tweaking and with little pre/postprocessing.

The overall metaprocess followed by our IDA is shown in Fig. 4. The user provides data, metadata, goals, and desiderata. Then, the IDA composes the set of valid DM processes, according to the constraints imposed by the user inputs, the data, and/or the ontology. This composition involves choosing induction algorithm(s), and appropriate pre and postprocessing modules (as well as other aspects of the process, not considered in this paper). Next, the IDA ranks the suitable processes into a suggested order based on the user's desiderata. The user can select plans from the suggestion list, hopefully aided by the ranking. Finally, the IDA will produce code for and can execute (automatically) the suggested processes on the selected data.

3 Enumerating Valid Data Mining Processes

Our first claim is that an ontology-based IDA can enumerate DM processes useful to a data miner. We support our claim in two ways. First, we describe how the ontology can enable the composition of only valid plans. Second, we describe process instances produced by our prototype (called IDEA), in order to provide evidence that they can be nontrivial. Later, we will describe how problem-specific elements can be incorporated into an IDA; for clarity and generality, first, we concentrate on domain-independent elements of the DM process. For example, when presented with a data set to mine, a knowledge-discovery worker (researcher or practitioner) generally is faced with a confusing array of choices [5]: Should I use C4.5 or naïve Bayes or a neural network? Should I use discretization? If so, with what method? Should I subsample? Should I prune? How do I take into account costs of misclassification?

3.1 An Ontology-Based Intelligent Discovery Assistant

Consider an example: A user presents a large data set, including both numeric and categorical data, and specifies *classification* as the learning task, along with the appropriate dependent variable. The IDA asks the user to specify his/her desired tradeoffs between accuracy and speed of learning. Then, the IDA determines which DM processes are appropriate. For our example task, decision-tree learning alone might be appropriate. Or, a decision-tree learner plus subsampling as a preprocess, or plus pruning as a postprocess, or plus both. Are naïve Bayes or neural networks appropriate for this example? Perhaps not by themselves. Not if the naïve Bayes implementation accepts only categorical attributes. Neural networks often accept only numeric attributes. However, preprocessing to transform the attribute type may enable their use.

The IDA uses the ontology to assist the user in composing valid and useful DM processes. Basing our design on AI planning [8] and semantic Web services [9], the prototype's ontology contains for each operator:

- a specification of the conditions under which the operator can be applied, including a precondition on the state of the DM process, its compatibility with preceding operators, and the inputs necessary for the execution of the algorithm,
- a specification of the operator's effects on the DM process's state and on the data,
- estimations of the operator's effects on attributes such as speed, accuracy, model comprehensibility, etc. (shown as heuristic indicators in Fig. 3), and
- a help function to obtain comprehensible information about each of the operators.

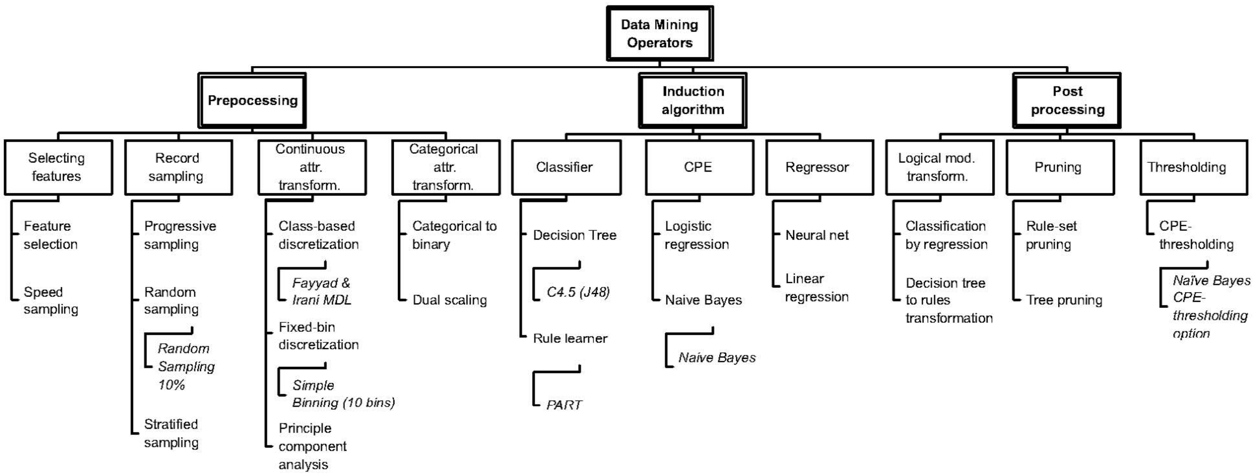

Fig. 5. The data mining ontology (partial view, the italicized leaf nodes were used in the ranking experiments).

In addition, the ontology contains schemata for generic problems such as target marketing. The schemata are represented internally as complex, decomposable operators with the same parameters as their simple counterparts. The only difference is that some of the steps within the complex operators might not be completely specified, opening a design space of subsolutions (Section 5 provides an example). The collection of all schemata is a case-base of processes thought or proven to be useful.

Fig. 5 shows a structural view of the prototype ontology, which, at the highest level, groups the DM operators into: preprocessing, induction, and postprocessing. Each group is further subdivided. At the leaves of this tree are the actual operators (selected examples chosen from Weka [5] are shown italicized). We constructed this prototype ontology by first considering the types of operators provided by data mining toolkits such as Weka [5]. We chose the three stages of the data mining process that have received the most automation—resulting in a nontrivial selection of operators. We then divided these into subcategories, focusing on operators that would be useful for cost-sensitive classification. Finally, we chose implementations that were available in Weka. After the ontological structure was in place, we called on our own knowledge to provide the operator specifics (preconditions, postconditions, etc., discussed above). Obviously, this ontology is not complete, even for cost-sensitive learning, and is limited by our knowledge. All of this was done before the experiments below were conducted, except as mentioned.

Based on the prototype ontology, we built a prototype IDA, which we call IDEA (**I**ntelligent **D**iscovery **E**lectronic **A**ssistant). Following our general framework for IDAs (see Fig. 4), IDEA first gathers a *task specification* for the DM process, analyzes the data that the user wishes to mine and extracts the relevant metadata, such as the types of attributes included (e.g., continuous, categorical). Using a GUI, the user then can complement the gathered information with additional knowledge about the data, and can specify the goal of data mining. IDEA's first core component, the *DM-process planner*, produces the set of valid processes and is described in Section 3.2.

A collection of valid DM processes typically will contain processes that are undesirable for certain user goals—e.g., sacrificing too much accuracy to obtain a model fast. IDEA's second core component, the *heuristic ranker*, ranks the valid DM processes using a combination of several heuristic functions. The GUI also allows the user: to specify tradeoffs (weights) between ranking functions, to sort the list of plans using any (weighted combination) of the rankings, to examine the details of any process plan, and to generate code for and to run the processes.

A final function of IDEA is to supply users with an interface to the ontology. Users can browse ontology entries with a tree-like hierarchy browser. To add new operators to the ontology requires adding a new element in the ontology tree and specifying its parameters. When adding the Weka ID3 tree-induction operator, for example, the user would first choose Decision Tree as an appropriate parent in the ontology. Then, the user would proceed to add the appropriate parameters. As a child of Decision Tree, the new ID3 operator would inherit some parameter values. The user must complement these parameters with the actual algorithm implementation, its call interface (for Weka's ID3: java-class weka.classifiers.trees.Id3 with the relevant method calls or a reference to a WSDL[3] file), and the heuristic parameters (such as `Speed = + 25`).

3.2 Enumerating Valid DM Processes: IDEA's Procedure

An IDA can produce a systematic enumeration of DM processes that will be useful to data miners, to help them avoid overlooking important processes. To enumerate (only) valid DM processes, IDEA performs a search of the space of processes defined by the ontology, constrained by the restrictions on operator application defined in the ontology. The structure of the search problem is amenable to more complex, AI-style planning, but for the results we present, a straightforward search was sufficient. IDEA constructs a specification of the sequence of DM operators (i.e., the DM process) that

3. WSDL (Web Services Description Language) is an XML format for describing Web services (or remote procedures), providing all information necessary to locate and call the service [10].

moves from the start state—the metadata description of the data set—to the goal state—typically, a prediction model with some desired properties. Starting with an empty process, at every state, it finds the applicable operators using the compatibilities, adds each operator (separately) to the partial process that brought it to the current state, and transforms the state using the operator's effects. Using the example above, in order to apply naïve Bayes, the current state must not contain numeric attributes, which would be the case after discretization. The planner would not apply discretization twice because after the first application, the state no longer would contain numeric attributes and, thus, the preconditions of discretization no longer would apply. The planner stops pursuing a given process when it has reached either the goal state or some "dead-end" state that will not lead to the goal state. The planner also can add complex operator schemata to any solution. Akin to hierarchical planning, it then must revisit all the nonspecified steps and treat them as planning problems themselves (see Section 5 for an example). The central difference from traditional, AI planning is that execution does not stop when a first viable solution is found. Instead, the search returns as many valid processes as possible, to aid users who are not able to express their preferences precisely or completely before seeing possible alternatives.

The constraints in the ontology are essential. If we use the ontology whose overall structure is shown in Fig. 5, give the goal of *classification*, and constrain the search only with the ordering of the logical groupings imposed by the prototype ontology (i.e., preprocessing precedes induction which precedes postprocessing), IDEA generates 163,840 DM processes. Adding the constraints imposed by the pre and postconditions of the operators (for example, neural networks require numeric attributes; decision-tree pruning can only apply to decision trees, etc.) IDEA produces 597 valid process instances—less than one-half of one percent of the size of the unconstrained enumeration. Adding metadata (e.g., the data set contains numeric attributes) or user desiderata (e.g., the user wants cost-sensitive classification) allows the enumeration to be constrained even further.

3.3 Enumerating Valid DM Processes: Example Enumerations from IDEA

The enumerations of processes produced by IDEA are not trivial. In many cases, they would be valuable not only to novice data miners, but even to experts.

Example 1. When IDEA is given the goal of producing a *cost-sensitive classifier for a two-class problem*, it produces an enumeration comprising 189 DM processes. The enumeration includes building a class-probability estimator and setting a cost-specific threshold on the output probability. It includes building a regression model and determining (empirically) an effective threshold on the output score. The enumeration also includes using class-stratified sampling with any classification algorithm (which transforms an error-minimizing classifier into a cost-minimizing classifier). Novice data miners certainly do not consider all these options when approaching a cost-sensitive problem. In fact, we are aware of no single published research paper on cost-sensitive learning that considers one of each of these types of option [11].

Example 2. When we give IDEA the goal of producing *comprehensible classifiers*, the top-ranked DM process is:

`subsample the instances` →
`feature selection` →
`use a rule learner` →
`prune the resultant rule set`.

Although comprehensibility is a goal of much machine-learning research, we are not aware of this process being used or suggested. This process is interesting because each component individually has been shown to yield more comprehensible models; why shouldn't the composition yield even more comprehensible models? As another DM process highly ranked by comprehensibility, IDEA suggests:

`build a decision tree` →
`convert tree to rules` →
`prune rule set`.

This also is a nontrivial suggestion: It is the process introduced by Quinlan [12] and shown to produce a combination of comprehensibility and high accuracy. Although the addition to the ontology of

`convert tree to rules`

was influenced by Quinlan's work, we did not "program" the system to produce this process. IDEA composed and ranked processes based on knowledge of individual operators. This is particularly valuable because the addition of a new operator to the ontology can have far-reaching effects.

Example 3. Consider the case where the user is interested in classification, but wants to get results fast. Does IDEA's enumeration contain particularly useful (fast) processes? Indeed, it suggests processes that use fast induction algorithms, such as C4.5 (shown to be very fast for memory-resident data, as compared to a wide variety of other induction algorithms [13]). It also produces suggestions not commonly considered [14]. For example, the enumeration contains plans that use discretization as a preprocess. Research has shown that discretization as a preprocess can produce classifiers with comparable accuracy to induction without the preprocess [15]; but with discretization, many induction algorithms run much faster. For example, as described by Provost and Kolluri [16], most decision tree inducers repeatedly sort numeric attributes, increasing the computational complexity considerably; discretization eliminates the repetitive sorting. IDEA's suggestions of fast plans also include plans that use subsampling as a preprocess. Researchers studying scaling up often do not consider subsampling explicitly, but, of course, it produces classifiers much faster—and, for large data sets, it may produce classifiers with comparable accuracies [17], [18].

TABLE 1
Sixteen Process Plans and Ranking

	steps (operator sequence)	*heuristic rank* credit-g speed	composition speed
Plan # 1	c4.5	13	13
Plan # 2	part	16	16
Plan # 3	rs, c4.5	2	5
Plan # 4	rs, part	8.5	10
Plan # 5	fbd, c4.5	12	11
Plan # 6	fbd, part	15	14
Plan # 7	cbd, c4.5	11	12
Plan # 8	cbd, part	14	15
Plan # 9	rs, fbd, c4.5	4	3
Plan # 10	rs, fbd, part	6.5	8
Plan # 11	rs, cbd, c4.5	5	4
Plan # 12	rs, cbd, part	6.5	9
Plan # 13	fbd, nb, cpe	8.5	6
Plan # 14	cbd, nb, cpe	10	7
Plan # 15	rs, fbd, nb, cpe	1	1
Plan # 16	rs, cbd, nb, cpe	3	2

Legend for operators used in plans	
abbrev.	*name/algorithm*
rs	Random sampling (result instances = 10% of input inst.)
fbd	Fixed-bin discretization (10 bins)
cbd	Class-based discretization (Fayyad & Irani's [1993] MDL)
c4.5	C4.5 (using Witten & Frank's [2000] J48 implementation)
part	Rule learner (PART, Frank & Witten [1998])
nb	Naïve Bayes (John & Langley [1995])
cpe	CPE-thresholding post-processor

Each plan is specified as the sequence of its composing steps (shown as operator abbreviations).

4 IDAs Can Produce Effective Rankings

Large enumerations of DM processes can be unwieldy. It is important to help the user to choose from among the candidate processes. Rankings of DM processes can be produced in a variety of ways. For example, static rankings of processes for different criteria could be stored in the system. Flexible rankings also are important—so that as new ontological knowledge is added, the system can take advantage of it immediately. IDEA produces rankings dynamically by composing the effects of individual operators. The ontology contains estimations of the effects of each operator on each goal. For example, an induction algorithm may be estimated to have a particular speed (relative to the other algorithms). Taking a 10 percent random sample of the data as a preprocess might be specified to reduce the runtime by a factor of 10 (which would be appropriate for algorithms for which runtime grows linearly with the amount of data). Correspondingly, sampling might be specified to reduce the accuracy by a certain factor (on average), and to increase the comprehensibility by a different factor (compare the study by Oates and Jensen [17]). For a given DM process plan, an overall score is produced as the composition of the functions of the component operators.

4.1 Details of Ranking Experiments

In order to provide a demonstration that IDAs can produce useful rankings, we coupled IDEA with a code generator that generates code for the Weka data mining toolkit[4] [5]. The system generates Java code for executing the plans, as well as code for evaluating the resulting models based on accuracy and speed of learning. We assess IDEA's ability to rank processes by speed and by accuracy because these are criteria of general interest to users and for which there are well-accepted evaluation metrics. Furthermore, one expects a rough trade off between speed and accuracy [13], and a user of an IDA may be interested in points between the extremes.

For the demonstrations in this section, we restricted the ontology to a subset for which it is feasible to study an entire enumeration of plans thoroughly. The ontology subset uses seven common preprocessing, postprocessing, and induction techniques (for which there were appropriate functions in Weka, see below). The experimental task is to build a *classifier*, and has as its start state a data set *containing at least one numeric attribute* (which renders some inducers inapplicable without preprocessing). Table 1 shows on the left the list of 16 valid process plans IDEA created for this problem; on the right is a legend describing the seven operators used.[5] Even this small ontology produces an interesting variety of DM-process plans. For example, the ontology specifies that naïve Bayes only considers categorical attributes, so the planner needs[6] to include a preprocessor that transforms the data. Although the ontology for the experiments is very small, the diversity of plans is greater than in many research papers.

In Table 1, the first column ranks the plans by the number of operators in the plan. This may be interesting to users who will be executing plans manually, who may be interested in minimizing fuss. We will not consider this ranking further except to reference plans by number. The *heuristic rank* columns of Table 1 show a pair of speed-based rankings computed by heuristics. The "credit-g" ranking is a static ranking created by running all the plans on one, randomly selected data set (viz., credit-g, not used for testing). A static ranking makes practical sense if the flexibility to add new operators is not of primary importance. Adding new operators (or otherwise

4. The choice of Weka was driven by the availability of a large number of suitable machine learning operators. Weka does have the drawback that, for the most part, it operates on in-memory structures making it unsuitable for exploration of some realistic large-scale data sets. In particular, the preprocessing steps, which often entail accessing large databases, should be handled a suitable database environment or within a full-scale data mining preprocessing environment like Mining Mart [19].

5. The last operator in Table 1, `cpe`, which places an appropriate threshold on a class-probability estimator, becomes a no-op for Naïve Bayes (`nb`) in the Weka implementation, because Weka's implementation of `nb` thresholds automatically.

6. This is not strictly true for the Weka implementation, for which naïve Bayes is augmented with a density estimator for processing numeric attributes. The Weka implementation could be considered naïve Bayes plus a particular numeric preprocessor.

TABLE 2
Data Set Names and Sizes

Dataset name	Size
heart-h	294
heart-c	303
ionosphere	351
balance-scale	625
credit-a	690
diabetes	768
vehicle	846
anneal	898
vowel	990
credit-g	1000
segment	2310
move	3029
dna	3186
gene	3190
adult10	3256
hypothyroid	3772
sick	3772
waveform-5000	5000
page	5473
optdigits	5620
insurance	9822
letter	20000
adult	32561

changing the ontology) changes the space of plans, in which case a static ranking would have to be updated or recomputed. The "composition" ranking was generated by a functional composition; the ontology specifies a base accuracy and speed for each learner, and specifies that all the preprocessing operators will reduce accuracy and will increase speed, by different amounts. The heuristic functions are subjective, based on our experience with the different data mining techniques and on our reading of the literature (e.g., [13]). The ranking functions were fixed before we began using Weka's particular implementations, with one exception: Because speeds differ markedly by implementation, we ran Weka on one data set (credit-g, again) to instantiate the base speed for the learning algorithms and the improvement factors for sampling and for discretization.

Our experiments compare the proposed (ex ante) rankings to (ex post) rankings generated by actually running the plans on the data sets. For the experiments, we used 23 data sets from the UCI Repository [20], each containing at least one numeric attribute. The data sets and their total sizes are listed in Table 2. Unless otherwise specified, for each experiment, we partitioned each data set randomly into halves (we will refer to these subsets as D_1 and D_2). We used 10-fold crossvalidation within D_2 to compute average classification accuracies and average speeds—which then are used to construct the actual (ex-post) rankings and to assess the quality of the ex-ante rankings. (We will use the D_1s, later, to construct auto-experimentation rankings; the $\{D_1, D_2\}$ partitioning ensures that all results are comparable.)

4.2 Ranking by Speed

Our first experiments examine whether the heuristics can be effective for ranking DM processes by speed. Since being able to rank well by speed is most important for larger data sets, consider the largest of our data sets: adult. Table 3

TABLE 3
Adult Data Set Rankings by Speed

Plan Name	credit-g ranking	composition ranking	D2 ("actual") ranking
Plan # 2	16	16	16
Plan # 6	15	14	15
Plan # 8	14	15	14
Plan # 1	13	13	13
Plan # 7	11	12	12
Plan # 4	9	10	11
Plan # 5	12	11	10
Plan # 14	10	7	9
Plan # 10	7	8	8
Plan # 12	7	9	7
Plan # 3	2	5	6
Plan # 13	9	6	5
Plan # 11	5	4	4
Plan # 9	4	3	3
Plan # 16	3	2	2
Plan # 15	1	1	1

shows the two heuristic rankings and the actual (ex-post) ranking based on the average runtimes for all the plans. The table is sorted by the actual ranking, and the table entries are the positions of each plan in each ranking (i.e., 1 is the first plan in a ranking, 2 the next, etc.). Both heuristics rank very well. For the credit-g ranking (on the adult data set) Spearman's rank-correlation $r_s = 0.93$ and for the composition ranking, $r_s = 0.98$ (recall that perfect rank correlation is 1, no correlation is 0, and a perfectly inverted ranking is -1).

Table 4 shows for all the domains the correlations between the rankings produced by the heuristics and the rankings based on the actual speeds. Here, the data sets are presented in order of increasing size (large ones toward the bottom). Highlighted in bold are the cases where $r_s > 0.5$ (all but the smallest data set).[7] Neither heuristic is superior, but both are effective; for both ranking heuristics, the average is approximately $r_s = 0.85$.

4.3 Ranking by Accuracy—Using Autoexperimentation

Our next demonstration examines whether the IDA can be effective for ranking DM processes by accuracy. Note that one would not expect to be able to do nearly as well at this task as for ranking by speed. Nevertheless, it would be helpful to be able to give users guidance in this regard, especially when a system proposes a process containing a component with which the user is not familiar—if the process were ranked highly by accuracy, it would justify learning about this new component. However, our attempt to use heuristic scores, similar to those that were successful for ranking by speed, did not produce particularly good accuracy rankings. Fortunately, an IDA can perform *autoexperimentation*, composing process plans and running its own experiments to produce a ranking of the plans by accuracy.[8] Although this may be the best possible ranking

7. The choice of 0.5 was ad hoc, but was chosen before running the experiment. Examining hand-crafted rankings with various r_s values seemed to indicate that 0.5 gave rankings that looked good.

8. This is not an option for speed rankings because the autoexperimentation process itself may be (very) time consuming.

TABLE 4
Spearman's r_s for Ranking Heuristics for Speed and Accuracy

	speed		accuracy
	credit-g ranking	composition ranking	auto-experiment.
heart-h	0.39	0.30	-0.06
heart-c	**0.62**	**0.59**	0.06
ionosphere	**0.80**	**0.70**	0.20
balance-scale	**0.82**	**0.81**	**0.55**
credit-a	**0.94**	**0.91**	**0.71**
diabetes	**0.55**	**0.64**	0.49
vehicle	**0.94**	**0.95**	**0.91**
anneal	**0.98**	**0.92**	**0.90**
vowel	**0.90**	**0.93**	**0.90**
segment	**0.89**	**0.92**	**0.92**
move	**0.90**	**0.95**	**0.87**
dna	**0.98**	**0.94**	**0.91**
gene	**0.92**	**0.95**	**0.88**
adult10	**0.97**	**0.97**	**0.86**
hypothyroid	**0.95**	**0.91**	**0.96**
sick	**0.95**	**0.89**	0.18
waveform-5000	**0.90**	**0.94**	**0.94**
page	**0.86**	**0.85**	**0.74**
optdigits	**0.89**	**0.87**	**0.84**
insurance	**0.95**	**0.93**	**0.84**
letter	**0.90**	**0.96**	**0.96**
adult	**0.93**	**0.98**	**0.86**
mean	**0.86**	**0.85**	**0.70**
median	**0.90**	**0.92**	**0.86**

method (albeit time consuming), even careful experimental evaluations of the accuracies of predictive models produce only estimations of the accuracy of the models on unseen data. The quality of the rankings of DM processes produced by such estimation will vary (e.g., by data set size), and for any particular domain must be determined empirically.

We now present an experiment to assess the effectiveness of such a procedure. For each domain, IDEA composed DM process plans and generated Weka code for the plans and for their evaluations via crossvalidation. For each data set, the crossvalidation was performed on data subset D_1 to produce an estimation of the accuracy that would result from running the plan on a data set from the domain. These accuracies were used to construct a ranking of the DM-process plans by accuracy for each data set. These rankings then were compared to the ranking produced on data set D_2 (identically to the previous experiments). Table 4 lists the resulting rank correlations in the rightmost column. As expected, the empirically determined rankings are considerably better for the larger data sets: averaged $r_s = 0.86$ for the data sets with >= 5,000 records.

4.4 Trading Off Speed and Accuracy

For large data sets autoexperimentation provides good accuracy rankings, but one pays a considerable runtime price as the data set size grows. What if a user is willing to trade off some speed for a better accuracy ranking, but does not have the time for full-blown autoexperimentation (i.e., running all the plans on all the data)? An alternative is to perform autoexperimentation on subsamples of the data to estimate the accuracy ranking for the full data set. We now demonstrate that our IDA can allow users to trade off quality of ranking for timeliness.

IDEA ran the process plans for the six largest data sets (each having 5,000 or more total records) on increasingly larger subsets of the data. Specifically, for each domain's D_1, we selected random subsets of $10\%, 20\%, \ldots, 100\%$ of the data. For each subset, IDEA performed crossvalidation to determine empirically an expected accuracy ranking, identically to the previous experiment. For this experiment, we consider only the eight DM-process plans that do not (already) contain random sampling. Fig. 6 plots the rank correlations as the size of the sample grows, and in bold shows the average rank correlation as size grows. As expected, the largest samples give better rankings than the smallest ones. For the 100 percent sample, all are above 0.5, and all but optdigits are above 0.8. On the other hand, for several of the data sets (page, adult, letter) the rankings with the 10 percent sample are not much better than random.

With one notable exception, the rank correlations become relatively stable when about half of the data have been seen. The optdigits curve is unusual: The rank correlations do not increase and do not become more stable as more data are used. Investigation shows that all methods perform so similarly relative to each other, even with small training sets (optdigits is "too easy," it turns out), that it is not possible to rank them meaningfully beyond a certain level. Of course, if all methods perform identically, then the ordering of the

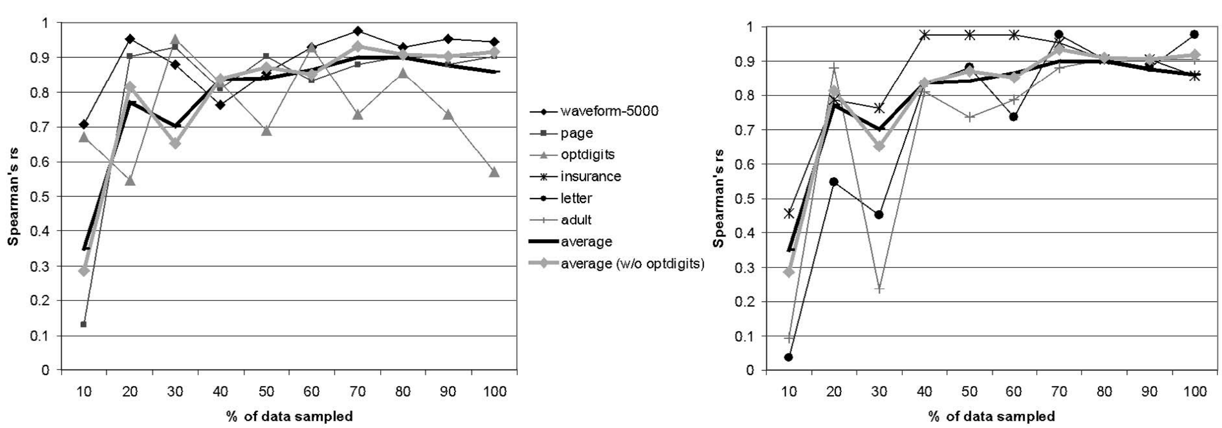

Fig. 6. Rank correlations and sample size. To improve readability, the results are divided into two graphs. The averages in both figures take all data sets (except optdigits, where noted) into account.

ranking does not matter. Fig. 6 also graphs the average without the optdigits data (bold and marked with a ◇), showing that the average performance is as desired (generally increasing, but with decreasing marginal benefits). That is, IDEA can perform autoexperimentation with sampling to produce good ranking estimates.

In sum, the results in this section demonstrate that it is possible for an IDA to produce effective rankings of generated processes by different desiderata (speed and accuracy), and to produce rankings that make tradeoffs between the two.

5 Demonstration with a More Complex DM Process

We now present the results of a final set of experiments, to demonstrate further the power of IDAs. The prototypical DM-process template that we used for the discussions and experiments above was straightforward. However, in real-world situations, the DM process can be more complex [21]. We assert that in such cases the potential value of an IDA is even greater because there is greater need for expertise in technique and process.

The data we use for our demonstration were the subject of the 1998 KDDCUP. The rationale for choosing the KDDCUP 1998 data set was threefold. First, the data set highlights the strengths of the planning-and-ranking approach: the combination of human insight about the problem and machine support for the systematic exploration of the design space. Second, it allows us to show the applicability of IDEA in the context of a more complex, cost-sensitive learning problem, rather than the straightforward classification problem used for the previous demonstrations. Finally, the data set has already been preprocessed extensively, making it suitable for our prototype, which concentrates on the building of the classification model, not on feature construction and selection. Even with the extensive preprocessing, the KDDCUP 1998 problem is not trivial.

The KDDCUP 1998 problem was to select a subset of customers to whom to mail solicitations, in order to maximize profit (revenues minus the cost of mailing). Participants built models from the training data, using a wide variety of different methods. To determine the winners, the organizers evaluated (on a separate test set for which the true answers were hidden) how much profit each team's model would have garnered. More specifically, KDDCUP 1998 was based on data from a fund-raising campaign undertaken by a national veterans association. The customer base was a set of individuals who donated in prior campaigns, and the goal was to select those from whom to solicit donations in the current campaign. Each observation in the data set represents an individual, and includes (for example) the response to the prior campaign. The training set from the competition consists of 95,412 records and the test set consists of 96,367 records. The mailing cost is $0.68 and the average donation is $15.60 with a range of $1-$200. The donation frequency is approximately 5 percent of the population. Using the default strategy of mailing to everyone, the average profit over the test set is $10,560. The (actual) results of KDDCUP 1998 are presented in Table 5. For our experiment, we use the variables used by Zadrozny and Elkan [22].

TABLE 5
Results of 1998 KDDCUP

Participants	Profit	%Gain
Urban Science	$14,712	39.32
SAS	$14,662	38.84
#3	$13,954	32.14
#4	$13,825	30.92
#5	$13,794	30.63
#6	$13,598	28.77
#7	$13,040	23.48
#8	$12,298	16.46
#9	$11,423	8.17
#10	$11,276	6.78
#11	$10,720	1.52
#12	$10,706	1.38
#13	$10,112	-4.24
#14	$10,049	-4.84
#15	$9,741	-7.76
#16	$9,464	-10.38
#17	$5,683	-46.18
#18	$5,484	-48.07
#19	$1,925	-81.77
#20	$1,706	-83.84
#21	($54)	-100.51

This was a challenging competition: the spread between the different competitors is quite large. Nine of 21 entries produced lower profits than did the default strategy of mailing to everyone. The last-place entry actually lost money. The winners achieved a 39 percent increase in profit over the default strategy. The winners are experts in this sort of data mining: Urban Science specializes in building models for target marketing (and, in fact, they also won the 1997 KDDCUP). In second place is SAS, who also have extensive experience with this sort of modeling. The competitors with the lower scores most likely applied data mining tools in the manner typical of data mining/machine-learning research. As we will demonstrate, the *straightforward* application of existing tools is insufficient for high-level performance on these data. However, the inclusion of application-specific, data mining-process knowledge is. As we will show, it is essential for IDEA to incorporate application-specific process knowledge. First, let us consider how IDEA performs without doing so.

We followed a methodology intended to mimic the algorithmic portion (i.e., not including feature construction and selection) of the process that KDDCUP competitors would have taken. Specifically, we create rankings of DM processes considering only the training set (estimating the profit that would be obtained). To assess the quality of a ranking, we calculate the "actual" profits on the test set. The 1998 KDDCUP focused on a problem of cost-sensitive classification: classify into one of two categories, solicit or do not, taking into account the cost of false positives (the mailing costs) and the cost of false negatives (the lost revenue). We use a larger set of induction algorithms than in the experiments above, but for clarity, for this experiment we do not consider pre and postprocessing explicitly.

Process **NN**: Create dummies → Neural Network → Classification by regression

Process **Lin**: Create dummies → Linear Regression → Classification by regression

Process **Log(CPE)**: Create dummies → Logistic Regression(CPE) → CPE-Threshholding

Process **NB(CPE)**: Discretization → Naïve Bayes (CPE) → CPE-Threshholding

Process **Rule(CPE)**: Rule Learner(CPE) → CPE-Threshholding

Process **DT(CPE)**: Decision Tree(CPE) → CPE-Threshholding

Fig. 7. DM processes generated for cost-sensitive classification.

Fig. 7 shows six DM processes generated for cost-sensitive classification. As mentioned above, a wider variety of learning algorithms (from Weka) is used here, and only one process with each algorithm is generated. Specifically, the first two processes produce regression models: process "NN" is the application of a neural network learner and process "Lin" is the application of linear regression. As mentioned in Section 3.3, regression models can be converted to cost-sensitive classification models by a postprocessor that chooses (by experimenting with the training data) an appropriate threshold on the predicted (output) value ("classification by regression"). Both of these algorithms require categorical variables to be preprocessed into a set of binary "dummy" variables. The last four processes use algorithms that create "class probability estimators," which give an estimation of the probability that a new example belongs to the class in question ("will donate"). Such a model can be converted to a cost-sensitive classifier with a postprocessor that chooses a threshold decision-theoretically, taking into account the misclassification costs. Process Log (CPE) uses logistic regression, which also requires preprocessing of categorical variables into dummies. Process NB(CPE) uses naïve Bayes, for which discretization is used as a preprocess. Processes Rule (CPE) and DT (CPE) build rule-based and decision-tree models, respectively; these do not require the preprocessing of numeric or categorical variables.

Table 6 shows the ranking of these processes by estimated profit, the actual profit calculated on the test set, and the resulting percentage gain over the default strategy of mailing to everyone. The profit was estimated by autoexperimentation (using crossvalidation, as above) on the training data. Note that, except for the neural network classifier, the ranking by estimated profit is perfect. Unfortunately, even without the error, the procedure would have placed only ninth (of 21) in the competition. What's worse, only one of the processes actually beats the default strategy of mailing to everyone. To be fair, this was a difficult problem for data miners not familiar with modeling for problems such as target marketing. Indeed, the participants in the contest were serious data mining researchers and tool vendors, and only half were able to do significantly better than the default strategy.

What did the winner(s) do differently? They did not use more complicated mining algorithms. Rather, *they used a different DM process*, one that is known by specialists to be particularly effective for target marketing. Specifically, as shown in Fig. 8, a class probability estimator (CPE) is built to estimate the probability of donation; separately, a regression model is built (from the donors in the training set) to estimate the amount to be donated conditioned on the presence of a donation. These two models are used in combination: the product of the two, for any individual, estimates his/her expected donation. If the expected donation is greater than the cost of the promotion to that individual, in this case $0.68, then a mailing should be sent. Otherwise, it should not.

Such process knowledge, in this case about how to combine techniques to form effective special-purpose DM processes, can be added to an IDA's ontology by specialists—subsequently to be brought to bear by others. The specialists can simply add the target marketing process as a problem-solving schema to the ontology. Note that there still is a large degree of freedom, even given such a process template. What type of learner should be used for class-probability estimation? What type of regressor? What type of pre/postprocessing is required? Using hierarchical planning, the IDA constructs DM processes within the

TABLE 6
Process Plans Ranked by Estimated Profit, Showing Actual Profit and Gain over Default Strategy

Plan	Rank	Profit	%Gain
NN	1	$6,919	-34.48
Lin	2	$11,968	13.33
Log(CPE)	3	$10,520	-0.37
Rule(CPE)	4	$9,924	-6.02
NB(CPE)	5	$9,538	-9.68
DT(CPE)	6	$8,496	-19.54

Legend for operators used in plans	
acronym	**name/algorithm**
DT	Decision Tree
Log	Logistic Regression
NB	Naïve Bayes
Rule	Rule Learner
Lin	Linear Regression
NN	Neural Network

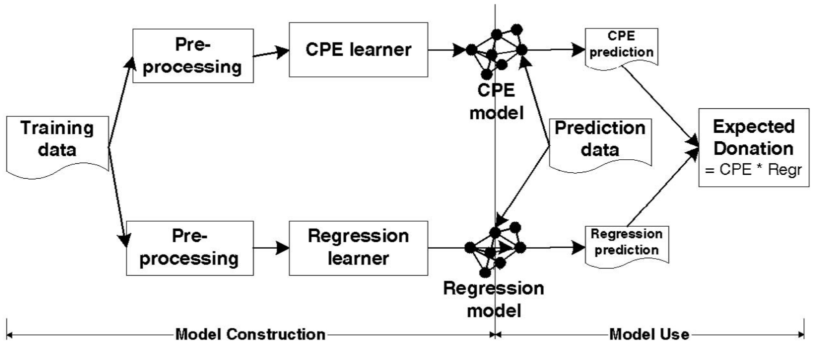

Fig. 8. Target marketing process.

constraints imposed by this template, in addition to the simpler, default template (which we used in previous sections).

For our final experiment, in addition to the six process plans based on the default (linear) DM process template, using the target-marketing template produces eight additional plans: the cross product of the available CPE learners (four) and the available regression learners (two). All the plans then are ranked by their estimated profit, produced via cross-alidation on the training set. If one plan were to be submitted to a contest such as the KDDCUP competition, it would be the highest-ranking plan. Of course, we have the luxury of examining the entire list.

The 14 process plans, ranked by crossvalidated estimated profit, are listed in Table 7 along with their test-set profits and the percentage gain (loss) over the default mailing strategy. The estimated ranking reflects the actual profit ranking quite well (with a couple notable glitches; Spearman's $r_s = 0.80$). Indeed, the range of gains is remarkably similar to the actual ranking of submissions to the contest (note that we excluded processes such as: (just) build a simple decision tree, which produces zero profit). The top-ranked plans indeed are competitive with the winners' submissions. The penultimate plan is the one used by the winning submission, and performs comparably in terms of profit. We did not expect the IDA to perform this well, because we figured SAS and Urban Science must have left some tricks up their sleeves (e.g., proprietary twists on the modeling algorithms). The top-ranked process actually would have beaten the winning submission.

These results illustrate not only the power of the IDA generally to enumerate and to rank processes effectively, but also the power of the IDA as a knowledge-sharing device. If one specialist includes knowledge about the target-marketing process, and another includes knowledge about neural networks, and yet another includes knowledge about logistic regression, other users would benefit from the IDA's composition of these to form a top-performing DM process.

6 Related Work

An IDA provides users with nontrivial, personalized "catalogs" of valid DM-processes, tailored to their task at hand, and helps them to choose among these processes in order to analyze their data. We know of little work that directly studies IDAs for the overall DM-process, although some have argued that they are important [23], [24]. There is, however, quite a long tradition of work that addresses some of the same goals (such as recommending and ranking) or using similar techniques (e.g., planning, auto-experimentation, and the use of ontologies) for recommending and for ranking individual induction algorithms.

6.1 The Use of IDAs

Especially in the European community, researchers have argued for the importance of IDAs. Morik [24], for example,

TABLE 7
Process Plans Ranked by Estimated Profit, Showing Actual Profit and Gain over Default Strategy

Plan	Rank	Actual Profit	%Gain
Log(CPE) + NN	1	$14,914	41.23
Log(CPE) + Lin	2	$14,778	39.95
Rule(CPE) + NN	3	$13,672	29.47
Rule(CPE) + Lin	4	$13,456	27.42
DT(CPE) + NN	5	$11,055	4.69
NN	6	$6,919	-34.48
DT(CPE) + Lin	7	$10,843	2.68
Lin	8	$11,968	13.33
Log(CPE)	9	$10,520	-0.37
NB(CPE) + NN	10	$10,070	-4.64
RULE(CPE)	11	$9,924	-6.02
NB(CPE)	12	$9,538	-9.68
NB(CPE) + Lin	13	$10,113	-4.23
DT(CPE)	14	$8,496	-19.54

Legend for operators used in plans	
acronym	name/algorithm
DT	Decision Tree
Log	Logistic Regression
NB	Naïve Bayes
Rule	Rule Learner
Lin	Linear Regression
NN	Neural Network
CPE	Class Prob. Estimator

proposes the use of a case-based repository to store successful chains of preprocessing operators.[9] Since preprocessing chains are partial DM processes, the insights gained should complement our work and, ideally, could be integrated with a system such as IDEA. The MetaL project[10] has as one of its foci IDA-like systems; we are not aware of any existing system that uses background knowledge and/or experimentation to compose and rank DM *processes*, although Brazdil argues that it is important to do so [23].

The only implemented IDA-like system we are aware of was presented by Engels et al., who describe a user-guidance module for DM processes called CITRUS [25], [26], [27], [28]. In particular, the user-guidance module uses a task/method decomposition [29] to guide the user through a stepwise refinement of a high-level DM process, in order to help the user to construct the best plan using a limited model of operations. Finished plans are compiled into scripts for execution. The system is implemented by extending SPSS Inc.'s Clementine® system, which provides a visual interface to construct DM-processes manually.

This work is similar to our approach as it provides the user with assistance when constructing DM processes, and uses AI planning techniques. In contrast, our approach is based on two notions that have led us in a different direction. First, even with a well-specified goal, it is very difficult to discern the one best plan because the results of running data mining methods are difficult to predict. Second, users' goals and desired tradeoffs often cannot be specified easily or completely at the onset of an investigation. This is because many desiderata are tacit and difficult to specify precisely (e.g., one may have an aversion to certain representations, based on experience with the domain experts). Moreover, knowledge discovery is an exploratory process; users must be given as much flexibility as possible. An IDA presents the user with many valid plans to choose from and helps him/her to choose among them, via rankings based on different criteria (and on combinations thereof). The user has no obligation to choose the highest-ranked plan in any given ranking—all of the plans in the ranking will be valid.

Akin to the approach of Engels and colleagues, Buntine et al. [30] introduce a method that generates data-analysis programs using program synthesis based on a declarative specification of the data-analysis problem. The declarative specification is a generalized Bayesian network. This approach is similar to ours in that it composes new knowledge discovery programs from a declarative problem specification. It differs from our approach in that it attempts to synthesize the one best program for data analysis (based on an optimization specification), rather than to provide the user with a series of options and help with choosing among them. The use of a deductive reasoning system for process synthesis is attractive because it allows guiding the planning process using new declarations rather than changing the planner.

9. See http://www-ai.cs.uni-dortmund.de/FORSCHUNG/PROJEKTE/MININGMART/index.eng.html.

10. MetaL stands for "MetaLearning," the process of learning models of the performance of learning algorithms as a function of characteristics of data sets; see http://www.cs.bris.ac.uk/Research/MachineLearning/metal.html.

6.2 Projects with Related Goals: Recommending and Ranking

A variety of research projects address issues regarding recommending/selecting optimal induction algorithms (rather than processes) and ranking induction algorithms. The knowledge generated from such projects could help to populate an IDA's ontology, as well as to inform the construction of more advanced functions for ranking processes. The MLT-Consultant [31] was an early system. It used a MYCIN-style knowledge base [32] with a Hypertext-based GUI to recommend to a user an algorithm to choose (from a machine-learning library). Several projects have since studied the selection of individual induction algorithms or subcomponents of algorithms based on certain forms of background knowledge. For example, Brodley [33] chooses subcomponents to form a hybrid decision tree, based on expert knowledge of algorithm applicability. The StatLog project[11] [34] investigated which induction algorithms to use given particular circumstances. Brazdil et al. [35], Gama and Brazdil [36], and others, use metarules drawn from experimental studies, to help predict which algorithms will be better; the rules consider measurable characteristics of the data (e.g., number of cases, number of attributes, kurtosis). This notion of "metalearning" is the basis for the MetaL project, mentioned above. Finally, Hilario and Kalousis [37] use a case-based system to advise users regarding which induction algorithm (and its respective parameter settings) to choose given a particular data mining task. Knowledge about the relationships between techniques and characteristics of data sets would fit well with the general notion of IDAs, but we have not (yet) incorporated any.

A different tradition of metalevel systems for data mining [38], sometimes called "automatic bias selection," involves the selection of one of the following, based in part on feedback from the performance of the learner: vocabulary terms, the induction algorithm itself, components of the induction algorithm, parameters to the induction algorithm [39]. Bias-selection work generally assumes the goal is accuracy maximization, but also applies to other desiderata [40], [41].

Addressing the need for improved ranking methods, several research projects have studied the use of experimental comparison to rank individual induction algorithms. Brazdil [23] summarizes some prior methods. This work is closely related to our ranking of DM processes (especially since one may put a conceptual box around a DM process and call it an induction algorithm, although this obscures important issues regarding the composition of processes). Brazdil and Soares have studied the ranking of individual induction algorithms, based on (functions of) their performances on previously seen data sets [42], [43]. They compare various methods for ranking, which perform comparably, and they consider ranking combining accuracy and speed.

6.3 Projects Using Similar Techniques: Landmarking, Planning, Knowledge Management, and Ontologies

As we have seen, many of the component methods necessary for building IDAs have been the subject of recent study. Several researchers have studied the notion of using fast processes (of different sorts) to help estimate the

11. See http://www.ncc.up.pt/liacc/ML/statlog/index.html.

performance of less efficient ones. Pfahringer et al. [44] and Fürnkranz and Petrak [45] provide overviews of such "landmarking" techniques. In particular, Petrak [46] shows the effectiveness of using subsamples from the data set in question to predict which learning algorithm will yield the lowest error on the entire data set; the technique works remarkably well—although it should be noted that for large data sets often one can achieve high accuracy with a surprisingly small subset of the data (compare progressive sampling [14]). On the other hand, the relative performance of algorithms can change markedly with the amount of data [18].

Statistical expert systems (compare [47], [48]) provide statistical advice. Most of the systems we are aware of base their advice on a statistical strategy, which is defined by Olford and Peters [49] as "... the reasoning used by the experienced statistician in the course of analysis of some aspect of a substantive statistical problem" (p. 337). Typically, those strategies are hand-coded to contain the multiple analysis alternatives of different problems such as regression analysis [50]. They help to guide the analysis of data by a human, to inform about which steps are likely to work next, and to allow direct execution. In contrast to our approach they do not offer support for a systematic exploration of the design space of possible processes (beyond the hand-coded strategies) nor for their relative rankings. One notable exception is the TESS system [51]. TESS is similar to our IDAs in that it allows the user to explore the search space of regression approaches using a heuristically guided search procedure. However, it predefines the set of approaches (specified using a tree structure), rather than using on an ontology-based planner.

St. Amant and Cohen [4] study intelligent, computer-based support for open-ended, statistical/exploratory data analysis. While focusing on somewhat different application areas, both their approach and ours employ mixed-initiative planning, where an AI-planner proposes different courses of action. The two approaches differ in how the human and the machine share control of the process. Statistical expert systems focus on step-by-step guidance, where the user can evaluate each step and get advice on what to do next. Our approach, on the other hand, presents the user with all possible plans and forecasts of their (relative) performance, allowing the user to choose one (or more) of the plans, run it, and then rerun the system based on insights gained. This latter approach is better suited in a domain (like knowledge discovery) where algorithms may run for extended periods of time. It may be worthwhile to create a hybrid approach that combines step-by-step guidance with overall planning allowing for the support of both types of data analysis.

Implementing Morik's [24] proposition mentioned briefly above, the Mining Mart project [19] stores best-practice cases of preprocessing chains that were developed by experienced users. The project has developed a data mining workbench, which allows users to draw on a case base to develop new data mining processes. Mining Mart is similar to our approach, in that it provides process-oriented discovery assistance to users based on an operator metamodel [52] and a case-base. It differs in that it does not provide users any planning facility or active support while choosing among the cases stored in the database. Finally, even though the Mining Mart metamodel is richer than the one we chose (it describes not only the operators but also captures metadata about the data set), it does not seem to take advantage of inheritance features, which can vastly simplify the implementation and engineering of an ontology. Our approach is complimentary to Mining Mart and advantage could be had from combining the strengths of both approaches.

Kerber et al. [53] document the DM process using active links to DM processes (that have been visually programmed) and to the rationale for major design choices. They collect these descriptions in a repository. This approach facilitates the reuse of DM processes, resulting in a knowledge management system for DM processes. It is complementary to our approach, as it emphasizes the documentation and retrieval of past knowledge, which could be integrated well with our notion of active support as represented by IDAs.

The only work of which we are aware that uses an explicit ontology within a metalevel machine-learning system is described by Suyama and Yamaguchi [54]. As far as we can tell, this system uses the ontology to guide the composition, by genetic programming, of fine-grained induction algorithm components.

7 Discussion, Limitations, and Future Work

We have argued for a systematic exploration of the design space of DM processes, without which users (even experts) seldom are systematic in their search of the DM-process space and, therefore, may overlook important, useful DM processes. Our IDA does not mimic the behavior of experts, who often use heuristics to materialize the hypothesis space to a "consideration set." Prepruning often leads experts to overlook excellent solutions, which lie outside of their consideration set [6].

For emphasis, we have discussed novice users and expert users. However, this is not a true dichotomy—there is a spectrum of expertise along which users reside. For the most novice, any help with DM process planning will be helpful. For the most expert, an IDA could be useful for double-checking, and for automating previously manual tasks, as well as for suggesting additional processes. For others along the expertise spectrum, IDAs will have both types of benefits. Furthermore, even among experts, different users have different expertise: a data miner trained in the statistics community and a data miner from the machine-learning community can be experts and novices with respect to different methods. An IDA may help to educate any user. For example, when the system produces a highly ranked plan that a user had not considered previously, the user can examine the ontology, and become educated on some new aspect of the DM process.

A unique benefit of an explicit ontology is the synergy it can provide between teams of users. If users contribute to the ontology, other users instantly receive the benefit of their contributions. Thus, an IDA may exhibit what economists call *network externalities* or *network effects*: the value to each user increases as the "network" grows. An IDA becomes more valuable to each user as the number of contributing users grows. All users get the benefit of each contributor's work automatically. No single member must be expert in all data mining technology.

Consider the following example of network effects in action. Jill is a member of a large team of data miners, with several on-going projects. While reading the statistics literature she discovers a technique called *dual scaling* [55], a preprocessing operator that transforms categorical data into (scaled) numeric data, in a manner particularly useful for classification. Jill codes up a new preprocessor (call it DS) and uses it in her work. Such discoveries normally are isolated; they do not benefit a team's other projects. However, consider what happens if Jill simply adds DS into the IDA. When another team member, Jack, uses the system, DM-process plans may be generated that use DS (when appropriate). In some cases, these plans will be highly ranked (when DS is likely to do a good job satisfying Jack's criteria). In such cases, Jack could experiment with DS immediately, or could read about it (using the documentation Jill added), or could follow pointers to the literature, or could call Jill directly and talk to her about it. Thereby, the tool brings to bear shared knowledge in the context of a particular need.

While we have provided no true experimental support for this assertion (adding the target-marketing template did greatly improved the performance of IDEA for the KDDCUP-1998 problem), empirical studies of the social aspects of knowledge sharing provide support for analogous claims in different application domains. Pentland [56], for example, shows how workers in a software hotline use a shared database as a central knowledge-sharing tool to become more effective as a team. Ackerman and Mandel [57] show how a knowledge-sharing tool helps astrophysicists learn from each other how to perform specific data analysis tasks.

We are not suggesting automating the DM process totally. In contrast, intricate user interaction is critical to successful discovery. We have shown that it is possible to provide automated, knowledge-based *assistance* for certain aspects of DM process design. We only have covered a few aspects so far, and for the most part only in the prototypical linear process. For example, our current prototype does not produce cyclic processes and our code generator does not yet produce code for more-complicated components, such as iterative feature selection [58] (e.g., around a subprocess), wrappers for parameter tweaking [41], progressive sampling [14], or the combination of the results of multiple plans using procedures such as Bayesian model averaging [59]. As new components are added, the space of DM processes will grow, and more knowledge or interaction may need to be brought to bear than is evident in the demonstrations we have provided here. On the other hand, this difficulty faces human data miners as well as IDAs, and the result seems to be that, even expert humans end up using only a small set of tools: those with which they are familiar. Even a moderately effective IDA would expand this set.

Our experiments with rankings serve to demonstrate that valid processes can be ranked effectively. As stated above, we have not yet studied the production of rankings in depth. Our IDA ranks the enumerations by characteristics such as speed, accuracy, and model comprehensibility. Some of those desiderata, such as speed and accuracy, have clear objective measures. Others are highly subjective. A statistician, for example, might rate a logistic regression equation as being very comprehensible, whereas a manager might not. A decision tree, on the other hand, might have the opposite result. Such preferences could be entered directly into a user-specific ontology, or could be discovered by the IDA using relevance feedback.

The related work on ranking induction algorithms should be very helpful for designing IDAs, but also provides important caveats. For example, our use of the Spearman rank-correlation coefficient in effect weights equally all positions in a ranking. However, for our purposes, the processes near the top of the ranking probably would be the critical ones (especially given a large number of generated process plans). Soares et al. [43] introduce a weighted modification to Spearman's coefficient, that takes into account position in the ranking. This same group of researchers also points out other challenges in comparing rankings, stemming from the fact that the "ideal" ranking typically is based only on estimates of the true error rates [42], [43].

Ranking DM processes using autoexperimentation raises the concern of multiple comparisons problems. Once the ontology is large, process-level overfitting may become an issue. This is a fundamental problem of manual or automatic search for good data mining processes, but is exacerbated by autoexperimentation.

We only have considered here parts of the process that are relatively well understood. Preprocessing existing variables, induction algorithms, and postprocessing learned models have received considerable attention in the literature. Other parts of the process are not as well understood or documented. For example, although feature construction has received research attention for years, our understanding of when and how to use it effectively pales in comparison with our understanding of these other parts of the process. Consider the KDDCUP 1998 problem we presented above. We side-stepped the issue of feature construction, which (we assume) was crucial to success in the competition. Does enough knowledge exist to provide an IDA with an ontology that will be effective to assist a user with feature construction? To our knowledge, this has yet to be shown convincingly. However, if generally effective methods or problem-specific heuristics exist, an IDA should be able to incorporate them. Additionally, our current prototype does not rely on detailed metadata (beyond attribute type). IDAs ought to use detailed metadata to restrict search as well as to inform the ranking, for example, by extending the findings of the Statlog and METAL projects to DM processes. We also have assumed that the user will perform the selection of the discovery task(s) to perform. A separate task is intelligent assistance for the selection of discovery tasks. This typically is ignored in discussions of the knowledge discovery process, but was addressed in early knowledge discovery work by Lenat [60] and recently by Livingston et al. [61].

Finally, although studies such as this are necessary for the development of useful IDAs, we also need well-designed (and executed) user studies to assess whether IDAs actually are effective in helping real data miners. Such studies could also provide indications of which features of IDAs are most effective in supporting the knowledge discovery process and, therefore, provide guidance for further improvements of IDAs.

8 Conclusion

Both novices and specialists need assistance in navigating the space of possible DM processes. We presented an ontology-based IDA, arguing that it can generate valid, nontrivial, and sometimes surprisingly interesting DM-process instances. Further, we have given empirical evidence that it is possible for IDAs to rank process instances effectively by speed and by accuracy, and have argued that they could rank by model comprehensibility, albeit subjectively. Finally, we have argued that IDAs can be particularly useful as a knowledge-sharing environment for teams of data miners, creating network effects wherein the tool becomes increasingly valuable as it gets more and more contributing users.

The knowledge discovery process has been a key concept in the field of KDD for more than a decade, but very little research addresses it explicitly. After having undertaken this work, we understand better why. Treating the DM process requires a tremendous breadth of knowledge of research and practical technique. Even most researchers know only a fraction of what is necessary to do a comprehensive job of building an ontology (and, we certainly have mistreated certain topics, although we have been careful). In retrospect, we believe even more strongly that in order for research on the knowledge discovery process to advance, systems like IDAs are essential—they document and automate parts of the process that are better understood, in order for research to concentrate on the large areas that are not well understood.

Acknowledgments

The authors would like to thank Yan Mao for her assistance with the implementation of the prototype system, IBM for a Faculty Award, and Carlos Soares and our reviewers for extensive comments on a prior draft. We are indebted to the authors of Weka, for providing the free data-mining toolkit, to the librarians of and contributors to the UCI Repository, and to the organizers and participants of the 1998 KDDCUP.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *Comm. ACM*, vol. 39, pp. 27-34, 1996.

[2] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0: Step-by-Step Data Mining Guide," SPSS Inc., 2000, http://www.crisp-dm.org/CRISPWP-0800.pdf.

[3] T. Senator, "Ongoing Management and Application of Discovered Knowledge in a Large Regulatory Organization: A Case Study of the Use and Impact of NASD Regulation's Advanced Detection System (RADS)," *Proc. Sixth Int'l Conf. Knowledge Discovery and Data Mining (KDD 2000)*, pp. 44-53, 2000.

[4] R. St. Amant and P.R. Cohen, "Intelligent Support for Exploratory Data Analysis," *J. Computational and Graphical Statistics*, vol. 7, pp. 545-558, 1998.

[5] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Francisco: Morgan Kaufmann, 1999.

[6] K.T. Ulrich and S.D. Eppinger, *Product Design and Development,* third ed. Boston: McGraw-Hill/Irwin, 2004.

[7] R. Kohavi, C.E. Brodley, B. Frasca, L. Mason, and Z. Zheng, "KDD-Cup 2000 Organizers' Report: Peeling the Onion," *SIGKDD Explorations*, vol. 2, pp. 86-93, 2000.

[8] M. Ghallab, C. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "lPDDL—The Planning Domain Definition Language," New Haven, CT TR-98-003/DCS TR-1165, ftp.cs.yale.edu/pub/mcdermott/siftware/pddl.tar.gz, 1998.

[9] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng, "DAML-S: Semantic Markup for Web Services," *Proc. Semantic Web Working Symp.*, 2001.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language 1.1," *World Wide Web Consortium (W3C) Technical Note*, www.w3.org/TR/wsdl, 2001.

[11] P. Turney, "Cost-Sensitive Learning Bibliography," Online bibliography, NRC Inst. for Information Technology, Ottawa, Canada, 2001, http://members.rogers.com/peter.turney/bibliographies/cost-sensitive.html.

[12] J.R. Quinlan, "Simplifying Decision Trees," *Int'l J. Man-Machine Studies*, vol. 27, pp. 221-234, 1987.

[13] T.S. Lim, W.Y. Loh, and Y.S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms," *Machine Learning*, vol. 40, pp. 203-228, 2000.

[14] F. Provost, D. Jensen, and T. Oates, "Efficient Progressive Sampling," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining (KDD '99)*, pp. 23-32, 1999.

[15] R. Kohavi and M. Sahami, "Error-Based and Entropy-Based Discretization of Continuous Features," *Proc. Second Int'l Conf. on Knowledge Discovery and Data Mining (KDD '96)*, pp. 114-119, 1996.

[16] F. Provost and V. Kolluri, "A Survey of Methods for Scaling Up Inductive Algorithms," *Data Mining and Knowledge Discovery*, vol. 3, pp. 131-169, 1999.

[17] T. Oates and D. Jensen, "The Effects of Training Set Size on Decision Tree Complexity," *Proc. 14th Int'l Conf. Machine Learning (ICML '97)*, pp. 254-262, 1997.

[18] C. Perlich, F. Provost, and J.S. Simonoff, "Tree Induction vs. Logistic Regression: A Learning-Curve Analysis," *J. Machine Learning Research*, vol. 4, pp. 211-255, 2004.

[19] K. Morik and M. Scholz, "The MiningMart Approach to Knowledge Discovery in Databases," *Intelligent Technologies for Information Analysis*, N. Zhong and J. Liu, eds., Springer, pp. 47-64, 2003.

[20] C.L. Blake and C.J. Merz, "UCI Repository of Machine Learning Databases," Univ. of California, Dept. of Information and Computer Science, 2000, http://www.ics.uci.edu/~mlearn/MLRepository. html.

[21] N. Agrawal, "Urban Science Wins the KDD-98 Cup. A Second Straight Victory for GainSmarts," 1998, http://www.kdnuggets.com/meetings/kdd98/gain-kddcup98-release.html.

[22] B. Zadrozny and C. Elkan, "Learning and Making Decisions When Costs and Probabilities are Both Unknown," *Proc. Seventh Int'l Conf. Knowledge Discovery and Data Mining (KDD 2001)*, pp. 204-213, 2001.

[23] P.B. Brazdil, "Data Transformation and Model Selection by Experimentation and Meta-Learning," *Proc. 10th European Conf. Machine Learning (ECML '98): Workshop Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, pp. 11-17, 1998.

[24] K. Morik, "The Representation Race—Preprocessing for Handling Time Phenomena," *Proc. 11th European Conf. Machine Learning (ECML 2000)*, pp. 4-19, 2000.

[25] R. Engels, "Planning Tasks for Knowledge Discovery in Databases; Performing Task-Oriented User-Guidance," *Proc. Second Int'l Conf. Knowledge Discovery in Databases (KDD '96)*, pp. 170-175, 1996.

[26] R. Engels, G. Lindner, and R. Studer, "A Guided Tour through the Data Mining Jungle," *Proc. Third Int'l Conf. Knowledge Discovery in Databases (KDD '97)*, pp. 163-166, 1997.

[27] R. Wirth, C. Shearer, U. Grimmer, T. Reinartz, J. Schlösser, C. Breitner, R. Engels, and G. Lindner, "Towards Process-Oriented Tool Support for KDD," *Proc. First European Symp. Principles of Data Mining and Knowledge Discovery (PKDD '97)*, pp. 55-64, 1997.

[28] F. Verdenius and R. Engels, "A Process Model for Developing Inductive Applications," *Proc. Seventh Belgian-Dutch Conf. Machine Learning,* pp. 119-128, 1997.

[29] B. Chandrasekaran, T.R. Johnson, and J.W. Smith, "Task-Structure Analysis for Knowledge Modeling," *Comm. ACM,* vol. 35, pp. 124-137, 1992.

[30] W. Buntine, B. Fischer, and T. Pressburger, "Towards Automated Synthesis of Data Mining Programs," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining (KDD '99),* pp. 372-376, 1999.

[31] S. Craw, D. Sleeman, N. Graner, M. Rissakis, and S. Sharma, "Consultant: Providing Advice for the Machine Learning Toolbox," *Research and Development in Expert Systems IX: Proc. Expert Systems, 12th Ann. Technical Conf. British Computer Soc. Specialist Group on Expert Systems,* pp. 5-23, 1992.

[32] R. Davis, "Interactive Transfer of Expertise," *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project,* The Addison-Wesley series in artificial intelligence, B.G. Buchanan and E.H. Shortliffe, eds., pp. 171-205, Reading, Mass.: Addison-Wesley, 1984.

[33] C.E. Brodley, "Recursive Automatic Bias Selection for Classifier Construction," *Machine Learning,* special issue on bias evaluation and selection, vol. 20, pp. 63-94, 1995.

[34] D. Michie, D. Spiegelhalter, and C. Taylor, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[35] P. Brazdil, J. Gama, and B. Henery, "Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning," *Proc. European Conf. Machine Learning (ECML '94),* pp. 83-102, 1994.

[36] J. Gama and P. Brazdil, "Characterization of Classification Algorithms," *Proc. Seventh Portuguese Conf. Artificial Intelligence,* pp. 189-200, 1995.

[37] M. Hilario and A. Kalousis, "Fusion of Meta-Knowledge and Meta-Data for Case-Based Model Selection," Technical Report, Univ. of Geneva, Geneva UNIGE-AI-01-01, 2001.

[38] B. Buchanan, C. Johnson, T. Mitchell, and R. Smith, "Models of Learning Systems," *Encyclopedia of Computer Science and Technology,* J. Belzer, A.G. Holzman, and A. Kent, eds., pp. 24-51, New York: M. Dekker, 1975.

[39] D. Gordon and M. desJardins, "Evaluation and Selection of Biases in Machine Learning," *Machine Learning,* vol. 20, pp. 5-22, 1995.

[40] D. Tcheng, B. Lambert, S. Lu, and L. Rendell, "Building Robust Learning Systems by Combining Induction and Optimization," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI '89),* pp. 806-812, 1989.

[41] F.J. Provost and B.G. Buchanan, "Inductive Policy: The Pragmatics of Bias Selection," *Machine Learning,* vol. 20, pp. 35-61, 1995.

[42] P. Brazdil and C. Soares, "A Comparison of Ranking Methods for Classification Algorithm Selection," *Proc. 11th European Conf. Machine Learning and Data Mining (ECML 2000),* pp. 63-74, 2000.

[43] C. Soares, P. Brazdil, and J. Costa, "Improved Statistical Support for Matchmaking: Rank Correlation Taking Rank Importance into Account," *VII Jornadas de Classificao e Analise de Dados,* pp. 72-75, 2001.

[44] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, "Meta-Learning by Landmarking Various Learning Algorithms," *Proc. Seventh Int'l Conf. Machine Learning (ICML 2000),* pp. 743-750, 2000.

[45] J. Fürnkranz and J. Petrak, "An Evaluation of Landmarking Variants," *Proc. 12th European Conf. Machine Learning and Data Mining/Fifth European Conf. Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2001) Workshop on Integrating Aspects of Data Mining, Decision Support, and Meta-Learning,* pp. 57-68, 2001.

[46] J. Petrak, "Fast Subsampling Performance Estimates for Classification Algorithm Selection," Austrian Research Institute for Artificial Intelligence TR-2000-07, 2000.

[47] W.A. Gale, *Artificial Intelligence and Statistics.* Reading, Mass.: Addison-Wesley Pub. Co., 1986.

[48] D. Hand, "Statistical Expert Systems," *Chance,* vol. 7, pp. 28-34, 1994.

[49] R. Oldford and S. Peters, "Implementation and Study of Statistical Strategy," *Artificial Intelligence and Statistics,* W.A. Gale, ed., pp. 335-353, Reading, Mass.: Addison-Wesley, 1985.

[50] R. Oldford, "Computational Thinking for Statisticians: Training by Implementing Statistical Strategy," *Proc. 29th Symp. Interface,* 1997.

[51] D. Lubinsky and D. Pregibon, "DataAnalysis as Search," *J. Econometrics,* vol. 38, pp. 247-268, 1988.

[52] M. Scholz and T. Euler, Documentation of the MiningMart Meta Model (M4)," TR12-05, IST Project MiningMart, IST-11993, 2002.

[53] R. Kerber, H. Beck, T. Anand, and B. Smart, "Active Templates: Comprehensive Support for the Knowledge Discovery Process," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98),* pp. 244-248, 1998.

[54] A. Suyama and T. Yamaguchi, "Specifying and Learning Inductive Learning Systems Using Ontologies," *Working Notes from the Proc. 1998 AAAI Workshop Methodology of Applying Machine Learning: Problem Definition, Task Decomposition and Technique Selection,* pp. 29-36, 1998.

[55] S. Nishisato, *Elements of Dual Scaling: An Introduction to Practical Data Analysis.* Hillsdale, N.J.: L. Erlbaum Associates, 1994.

[56] B.T. Pentland, "Organizing Moves in Software Support Hot Lines," *Administrative Science Quarterly,* vol. 37, pp. 527-548, 1992.

[57] M.S. Ackerman and E. Mandel, "Memory in the Small: Combining Collective Memory and Task Support for a Scientific Community," *J. Organizational Computing and Electronic Commerce,* vol. 9, pp. 105-127, 1999.

[58] R. Kohavi and G.H. John, "Wrappers for Feature Subset Selection," *Artificial Intelligence,* vol. 97, pp. 273-324, 1997.

[59] J.A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volinsky, "Bayesian Model Averaging," *Statistical Science,* vol. 14, pp. 382-401, 1999.

[60] D. Lenat, "AM: Discovery in Mathematics as Heuristic Search," *Knowledge-Based Systems in Artificial Intelligence,* McGraw-Hill Advanced Computer Science Series, D. Lenat and R. Davis, eds., pp. 3-225, McGraw-Hill, 1982.

[61] G. Livingston, J.M. Rosenberg, and B.G. Buchanan, "An Agenda- and Justification-Based Framework for Discovery Systems," *J. Knowledge and Information Systems,* vol. 5, pp. 133-161, 2003.

Abraham Bernstein is an associate professor and heads the Dynamic and Distributed Information Systems Group in the Department of Informatics at the University of Zurich, Switzerland. Before joining the University of Zurich, he was an assistant professor in the Department of Information, Operations and Management Sciences at New York University's Leonard N. Stern School of Business and received the PhD degree from MIT's Sloan School of Management, where he worked with Professor Thomas W. Malone at the Center for Coordination Science. His research interests include the various aspects of supporting dynamic (intra and inter) organizational processes with a special focus on machine learning and the semantic Web.

Foster Provost is an associate professor and NEC Faculty Fellow in the Department of Information, Operations and Management Sciences at New York University's Leonard N. Stern School of Business. Professor Provost is editor-in-chief of the journal *Machine Learning* and a founding board member of the International Machine Learning Society. He is a member of the editorial boards of the *Journal of Machine Learning Research* (*JMLR*) and the *Journal of Artificial Intelligence Research* (*JAIR*). In 2001, he cochaired the program of the ACM SIGKDD Conference. He has applied machine learning techniques to a variety of business problems, including fraud detection and customer contact management.

Shawndra Hill received the BS degree in mathematics from Spelman College and the BSEE degree from the Georgia Institute of Technology. She is currently a doctoral candidate in the Department of Information, Operations and Management Sciences at New York University's Leonard N. Stern School of Business and a research intern in the Statistics Department at AT&T Labs Research. Her research interests included data mining and machine learning with networked data.