Open access • Journal Article • DOI:10.1145/2480741.2480748

A survey of intelligent assistants for data analysis — Source link

Floarea Serban, Joaquin Vanschoren, Jörg-Uwe Kietz, Abraham Bernstein

Institutions: University of Zurich, Katholieke Universiteit Leuven

Published on: 03 Jul 2013 - ACM Computing Surveys (ACM)

Topics: Leverage (statistics)

A Survey of Intelligent Assistants for Data Analysis

FLOAREA SERBAN, University of Zurich
JOAQUIN VANSCHOREN, Katholieke Universiteit Leuven
JÖRG-UWE KIETZ and ABRAHAM BERNSTEIN, University of Zurich

Research and industry increasingly make use of large amounts of data to guide decision-making. To do this, however, data needs to be analyzed in typically non-trivial refinement processes, which require technical expertise about methods and algorithms, experience with how a precise analysis should proceed, and knowledge about an exploding number of analytic approaches. To alleviate these problems, a plethora of different systems have been proposed that "intelligently" help users to analyze their data.

This article provides a first survey to almost 30 years of research on Intelligent Discovery Assistants (IDAs). It explicates the types of help IDAs can provide to users and the kinds of (background) knowledge they leverage to provide this help. Furthermore, it provides an overview of the systems developed over the past years, identifies their most important features, and sketches an "ideal" future IDA as well as the challenges on the road ahead.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; H.5.2 [**Information interfaces and presentation**]: User Interfaces; I.2 [**Artificial Intelligence**]: Applications and expert systems, Learning; I.5 [**Pattern Recognition**]: General, Design methodology, Applications

General Terms: Data mining, user support

Additional Key Words and Phrases: Intelligent assistants, automatic workflow generation

1. INTRODUCTION

Living in the 'Information Age', we are facing a deluge of digital data being generated at ever higher speeds. However, to make this data *useful* for decision makers, one first needs to make sense of it.

Several data analysis systems exist that simplify the analysis and management of data, such as IBM SPSS Modeler[1], RapidMiner[2], SAS Enterprise Miner[3], and Weka[4]. Each of them provides a large number of analysis techniques and facilitates the gathering, application, inspection, and evaluation of data analysis operators and their re-

[1] http://www-01.ibm.com/software/analytics/spss/products/modeler/
[2] http://rapid-i.com/content/view/181/190/lang,en/
[3] http://www.sas.com/technologies/analytics/datamining/miner/
[4] http://www.cs.waikato.ac.nz/ml/weka/

This work was supported by the European Community 7^{th} framework ICT-2007.4.4 (No 231519) "e-Lico: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science", and is part of the programme of BiG Grid, the Dutch e-Science Grid, which is financially supported by the Netherlands Organisation for Scientific Research, NWO.
Author's addresses: F. Serban, J.-U. Kietz and A. Bernstein, University of Zurich, Department of Informatics, Dynamic and Distributed Information Systems Group, Binzmühlestrasse 14, CH-8050 Zurich, Switzerland; E-mail: {kietz—serban—bernstein}@ifi.uzh.ch; J. Vanschoren, Universiteit Leiden, Leiden Institute of Advanced Computer Science (LIACS), Niels Bohrweg 1, 2333CA Leiden, The Netherlands, and KU Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium. E-mail: joaquin@liacs.nl.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

sults. Most of these systems, however, lack any kind of guidance as to which techniques *can* or *should* be used in which contexts. Consequently, novice analysts are typically completely overwhelmed. They have no idea which methods can be confidently used, and often resort to trial and error. Given that the formulation of precise guidelines is often difficult or even impossible [Schaffer 1994; Wolpert 2001], specialists rely on years of accumulated tacit experience, which is hard to express explicitly [Nonaka and Takeuchi 1995]. Aggravating the problem is the ever-increasing number of operators [Hand 1997] that few experts can keep up with. Analysis of the entries from the KDD-cup (a data analysis competition) showed that the majority of experts use no more than two methods: a decision tree learner and their own induction method [Kohavi et al. 2000]. Obviously, *there is a dire need to support both experts and novices in their data analysis tasks*.

Traditionally, this kind of support is provided by experts/consultants. These are, however, often not available and also mired by the growth of methods. To address this issue, a number of researchers have proposed the use of automated advising systems (e.g., [Engels 1996; Bernstein et al. 2005; Charest et al. 2008; Žáková et al. 2010; Kietz et al. 2010]). The general idea is to build a system that advises users in all stages of a data analysis process [Fayyad et al. 1996] – essentially the provision of an automated Intelligent Discovery Assistant (IDA) [Bernstein et al. 2005].

Several attempts at developing IDAs have been made (e.g., [Amant and Cohen 1998b; Bernstein et al. 2005; Engels 1996; Gale 1986]). The main problem with research in the field of IDAs is that it is distributed over many disciplines (such as statistics, exploratory data analysis, planning, and Data Mining (DM)), and addresses different kinds of support situations (such as support for the whole KDD-process, automatic workflow generation, workflow design support, etc.). Furthermore, the IDA's developed up to now rely on many types of different background knowledge (such as operator applicability, operator performance, meta-data, etc.).

The first goal of this survey is to identify which types of support these various IDAs offer to the data analyst (Section 2). Second, it surveys the kinds of (background) knowledge that these IDAs rely on in order to provide this support (Section 3). Having defined these two central dimensions for categorizing IDAs, this survey then continues to list the types of IDAs found in an exhaustive literature search (Section 4) which, in turn, sets the stage for a thorough comparison of IDAs in light of the defined dimensions and the identification of limitations and missing features in Section 5. On the basis of these findings, the survey outlines a vision for an ideal future IDA in Section 6, and finally presents some summarizing conclusions in Section 7.

The main difference between the current survey and the previously published surveys on meta-learning [Vilalta and Drissi 2002b; Smith-Miles 2008] is, firstly, that we chose a more general approach by focusing on data analysis and not merely on meta-learning. Secondly, we identify and discuss the types of available/missing user support and link it to the background knowledge available to these systems. Likewise, we differ from surveys on DM tools [Goebel and Gruenwald 1999; Mikut and Reischl 2011] by focusing on a specific type of tools, namely, those that provide *intelligent support* for improving the users' experience with the data analysis process.

Consequently, the main contributions of this paper are (1) the structuring of the problem of user support for data analysis along two dimensions substantiated in the literature, (2) a thorough survey and comparison of this cross-disciplinary and sometimes difficult to access research field, and (3) the identification of fruitful avenues for the development of intelligent, user-centered IDAs.

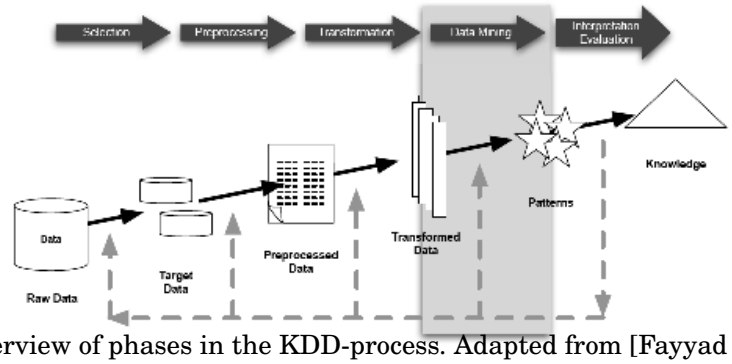

Fig. 1: Overview of phases in the KDD-process. Adapted from [Fayyad et al. 1996].

2. TYPE OF SUPPORT NEEDED

Fayyad et al. [1996] define the KDD-process—the Knowledge Discovery in Databases analysis process—which can serve as a general template for any data exploration. It structures the data exploration process along a set of transformations from the raw data to the actionable knowledge. As Figure 1 shows, this process is composed of the following steps:

Data selection. In this first step the relevant collections of raw data are identified.
Data preprocessing. Here the raw data is collected and cleaned, and the missing values and errors as well as noise are handled.
Data transformation. The preprocessed data is now transformed into a format suitable for DM. This ranges from converting complex data (e.g. images) into a set of useful features to dimensionality reduction techniques, feature selection as well as data sampling.
Data mining. In this step the DM or statistical modeling techniques are applied to find models or patterns.
Evaluation. Finally, the generated models are interpreted and/or evaluated.

Support for users analyzing the data could be incorporated in each of these phases, in a combination thereof, or even in its entirety. It can help the users with the construction of an analysis, with its execution, or with its understanding. In this section, we explore all these types of support developed by scientists over the past in order to gather the set of requirements for an IDA.

2.1. Support for a single step of the KDD-process

Some systems help the users to model a single step of the KDD-process by guiding them through the process of applying a specific operator. They provide hints and advice on selecting the right type of input data (e.g., nominal, scalar, ordinal) for a specific operator including parameter selection and tuning.
Each step of the KDD-process can be performed by several algorithms. The challenge, however, is to find the right one for the given data and analysis task at hand. Novices are usually overwhelmed by the plethora of approaches and tend to revert to only a few known techniques. But even experts are often unaware of suitable, less-known methods as shown in [Kohavi et al. 2000; Bernstein et al. 2005].
Moreover, each algorithm owns several parameters; therefore, it acts differently if diverse parameter values are chosen. Thus, one needs to find the appropriate values for the parameters to get the optimal output on a certain data set, since the data set influences the process of choosing the best matching parameters.

Lastly, some techniques have specific requirements on the form of data, as for example the ID3 tree induction algorithm uses only nominal attributes. Therefore, stepping back to the data preparation phase is often needed. Hence, support in choosing the adequate algorithm and parameters for a given analysis need is clearly desired.

2.2. Support for multiple steps of the KDD-process

A less common feature in IDA systems is the support for the overall KDD-process. Here the system provides useful help regarding the sequence of operators in the KDD-process as well as their parameters. Hence, it provides help in addressing the overall data analysis process from raw data to the actionable knowledge.

In order to be able to support the whole process, the system must have knowledge about the usage of each step in correlation to the other ones. More precisely, it has to know what operators can be applied in which situations as well as how to set the parameters of those operators. This knowledge is extracted from DM cook-books, such as the CRISP-DM [Chapman et al. 1999] or from experienced users.

Note that effective support for the overall process should go beyond the mere construction of correct processes. It may need to consider domain- and task-specific knowledge as well – a capability that has eluded AI-based approaches and required human creativity in analyzing data so far. The support for combining multiple steps (e.g., auto wiring, correction proposals, next operator recommendation) does not imply that they are planned fully automated.

2.3. Graphical design of KDD workflows

Graphical editing refers to enabling the user to interactively build the process manually: choosing the operators, and setting the right inputs/outputs and parameters. Some of these GUIs integrate intelligent techniques, such as auto-wiring, meta-data propagation, correctness checking before execution, and operator recommendation. Even if not, they can be viewed as a baseline for DM as it is used today and which has to be improved with intelligent assistance. They are often seen as being more user-friendly than textual representations of the KDD-process. Graphical editing systems allow the users to drag and drop operators and connect them via inputs/outputs (called ports). Such systems use either implicit data objects (no representation) or explicit ones. Often, several tabs are used for displaying information to the users as well as pop-ups, graphs, etc. In addition, different layouts and colors are employed for displaying the data as well as meta-data and its propagation.

2.4. Automatic KDD workflow generation

A special case of user support for multiple steps of the KDD-process is the automatic generation of workflows. Here the system provides the users with one or more KDD-processes that they need to execute in order to solve the DM task. The system automatically sets all inputs/outputs and parameters for each operator. This is especially useful for users who do not have a lot of experience with DM operators. Based on the data and a description of their task, the users receive a set of possible scenarios for solving a problem.

Automatic generation is a much more challenging problem than the previous ones since it requires knowledge about the order of the steps in the KDD-process. Usually the knowledge is extracted from DM cook-books and from expert data miners. In general, automatic generation involves a number of issues that need to be addressed:

— *How compatible is the generated workflow with user-modifications?*
  It would be useful if the user were able to make modifications to the generated workflows and execute them. The user feedback should be maintained.

— *How good are the generated KDD workflows?*
The quality of a workflow is defined both by its correctness in execution and its performance with respect to evaluation criteria such as accuracy and speed. In addition, the systems can either "just" list the best performing workflow or rank all of the constructed ones according to various evaluation criteria.
— *How many workflows are generated?*
Some systems generate only one solution (as in classical planning); others try to generate all possible solutions.
— *Which ones are generated and why?*
Does workflow generation follow certain rules or heuristics? How are these chosen?

Many practitioners and experts believe that the KDD-process cannot be fully automated since it is a creative process and the user has to constantly check the status of the workflow. At some stages of the workflow, only a human user can decide what is correct or not. Today's systems, for example, encounter major difficulties in identifying the correct data-type for some attributes (e.g., a number should be interpreted as either nominal or ordinal or scalar?).

2.5. Workflow checking/repair

Automatic workflow checking is not only an important feature of data analysis systems but also a recommended feature for IDAs. It checks a user-designed workflow for correctness and displays potential problems. For example, the system checks whether the port type is correct, whether the input data has the right type, etc. Additionally, another useful feature is the system's ability to repair a workflow or suggest fixes for generating correct workflows. The system can automatically connect the right ports, suggest what is wrong and what the user should do in order to solve the problems.

2.6. Task decomposition

As shown in knowledge acquisition, task decomposition structures simplify large processes [Chandrasekaran et al. 1992]. Some IDAs, hence, allow modeling KDD-processes using task decomposition primitives [Engels 1996]. Task descriptions can be reused, thus decreasing the development time and simplifying the process of decomposing a KDD task.

2.7. User design support

Additionally to providing a graphical editor, IDAs can display information about operators and their usage akin to tool-tips in modern Integrated Development Environments such as Eclipse. They allow users to easily obtain information about operators for solving problems or errors. Furthermore, some information can pertain to multiple steps: CRISP-DM [Chapman et al. 1999], for example, provides guidelines for structuring a KDD-process by reminding of what to do when and how.

2.8. Explanations regarding a decision or a result

Whilst IDAs may offer assistance in the KDD-process, it is important that they provide the rationale for the assistance. The rationale is the basic building block for aiding the users in sense-making [Russell et al. 1993] and allowing them to reason about the aid provided by reflecting on whether they wish to accept the help or proceed differently. Additionally, some IDAs also support the interpretation of results by automatically providing graphs thereof or choosing other appropriate representations.

2.9. Execution of operators/workflows

Some IDAs allow the users to execute workflows they created or which are generated automatically by the system. Not all IDAs implement their own operators; some only make use of existing statistical or ML libraries. Other IDAs inform the user about the usage of resources and provide estimations about the time needed to execute operators on data sets as well as the space used (i.e., the main storage for the processing and the disk-storage for the results).

For the systems that support automated generation of workflows, execution also includes experimentation in the design process for finding which workflows perform better for which data sets. The experimentation results are further used to provide workflow rankings by different criteria (i.e., accuracy, execution time, etc.).

2.10. DM experience level

Another attribute of IDAs that has been considered by Hand is the user's prior experience [Hand 1985]: *Who is actually using the IDA?* The users can be either DM experts or naïve users (experts in other fields like biology, genetics, etc) but have less knowledge about DM. Depending on the type of user considered, the system needs to provide different kinds of support.

Naïve users typically encounter two kinds of problems: First, since their understanding of DM concepts is limited, it is difficult to decide which DM operators to choose when analyzing their data. Second, it is difficult to determine which sequence of operators to apply as well as how to interpret the results obtained. Hence, users need to be led through the sequence of steps they have to apply, subsequently they need to be explained how the decisions were made and why a certain operator was selected. Above all, the system should protect the users from misuse, mistakes and misunderstanding of the DM terms.

Experts, on the other hand, have different needs and requirements. They already know how to use the main DM operators, how to modify their parameters, and how to gain a better performance. They rather need help in discovering new and more appropriate operators and in overcoming inertia at choosing the adequate approaches. In addition, the combination of operators with their parameters sometimes results in too many possible solutions, and therefore, it is impossible for a human to find the appropriate sequence. Hence, an IDA could help by suggesting workflows that were ranked according to different criteria. The experts can then compare their own solutions and validate their own decisions. Usually, experts do not want to be slowed down by being asked too many elementary questions or be led inflexibly through steps. They prefer to be able to skip support at any time or even modify a certain step. The ideal IDA should leave the expert to direct the decisions and provide explanations only on demand.

3. TYPES OF KNOWLEDGE

These various types of user support can only be offered by a system that has access to prior (background) knowledge on the involved data analysis techniques. This expertise should be stored in a machine-interpretable format so it can be automatically extracted and applied on new problems. A crucial aspect of designing an IDA thus lies in deciding *which* prior knowledge to save and *how* to store it. In this section, we categorize these types of expertise.

3.1. Available operators

First, a system must know about the different existing techniques that can be applied in the data analysis process. These techniques are most often represented as *operators*, which can be assembled in more complex data analysis workflows. Each IDA has a

certain 'registry' of operators : some handle operators across the whole KDD process, while other focus on only one of the KDD phases as shown in Figure 1. The operators, however, are usually not implemented in the IDA itself: they are most often part of an existing DM platform, and accessed through an application programming interface (API) or through web services.

3.2. Meta-data on the input dataset

To know which techniques can be applied, the system also needs information about the input data it receives. This information can range from simple properties, like the amount of attributes in tabular data or the amount of missing values, to complex information-theoretic properties, like the entropy of value distributions or the signal-to-noise ratio. Early IDAs used to inquire the (expert) user for such information, but most modern IDAs calculate all required characteristics automatically.

3.3. Meta-data on operators

Typically, the input data properties are linked to the operator properties in order to provide advice. We distinguish between two types of operator meta-data: *external* (black-box) properties, i.e. the operator's inputs, outputs, preconditions and effects (abbreviated as IOPE), and *internal* properties linked to operator's structure (e.g. parameters or model type) or performance (e.g. speed, accuracy, model complexity).

IOPE information is essential for knowing when to apply an operator. For instance, some modeling techniques take tabular data (input) and generate a predictive model (output). Moreover, certain techniques can only handle continuous values (precondition) or may be slow, resulting in longer execution times (effect). IOPE can also be used for checking the validity of a workflow, for visualizing workflow connections in a graphical interface, and even for supporting automatic workflow generation via AI planning. The IOPE information is most often stored in hardcoded form (through an API), as formal planning domain descriptions (e.g. STRIPS or PDDL [Fox and Long 2003]), in *ontologies* [Chandrasekaran and Josephson 1999], or in the Semantic Web Rule Language (SWRL) [Horrocks et al. 2004].

In addition to IOPE, many IDAs also store the meta-data related to the operator's performance, e.g. speed or model accuracy. Some IDAs even store detailed meta-data information of prior operator executions in a database. This meta-data can then be used to estimate the operator's performance based on similar cases, or to continuously update predictive models applied by the system (see below).

3.4. Predictive models

Some IDAs store and utilize *models* (e.g. rules) that suggest when to apply certain operators. For instance, if the data contains missing values, such models may advise the user to first impute those missing values in a preprocessing step, or suggest a modeling technique that is less sensitive to them. These models can come in the form of *expert rules* crafted by human experts or they can be more complex models that were automatically extracted from meta-data on previous data processing workflows (see Section 4.2). They can be expressed in human-readable form, e.g. in SWRL rules, as formal model descriptions (e.g. in PMML), or in custom formats that need to be loaded into a certain operator before being able to make predictions.

3.5. Case base

It is often useful to maintain a set of successful prior data analysis workflows so they can be adapted and reused in similar situations or simply serve as an inspiration to the data analyst. A knowledge base of such successful workflows is called a *case base* and it is typically maintained by experts who populate the case base with useful workflows.

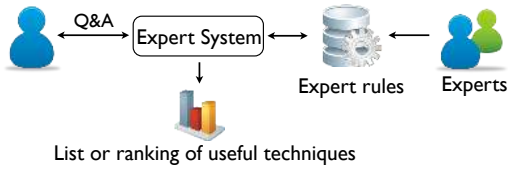

Fig. 2: The general architecture of expert systems

Furthermore, there exist collaborative platforms for sharing scientific workflows, like myExperiment [Goble et al. 2010], which allow users to share their workflows with other scientists, and collaborate on improving them.

Case bases represent a more implicit type of background knowledge: they form a collective 'memory' of successful solutions that may speed up the generation of new workflows and avoid pitfalls. The most similar, most successful prior cases are often retrieved using Case Based Reasoning (CBR), or by a simple keyword search.

4. TYPES OF SYSTEMS

This section provides an overview of the various IDAs developed to date. We focus on the systems that provide the user with intelligent guidance for designing workflows and will skip systems that merely offer tools but no specific guidance. We categorize these IDAs into five categories based on the core technique used to generate useful advice:

— *Expert systems* apply rules defined by human experts to suggest useful techniques
— *Meta-learning systems* automatically learn such rules from prior data analysis runs
— *Case-based reasoning systems* find and adapt workflows that were successful in similar cases
— *Planning systems* use AI planners to generate and rank valid data analysis workflows
— *Workflow composition environments* facilitate manual workflow creation and testing

4.1. Expert systems

The earliest systems designed for assisting users in data analysis were Expert Systems (ES); a high-level overview of which is shown in Figure 2. They were centered around a knowledge base of expert rules, which were hand-crafted by experts. These rules dictate which techniques should be used under which circumstances. For assessing which rules to apply, the system asks the user questions about the given problem. After that, it typically returns a list or ranking of recommended techniques.

4.1.1. Forerunners: Statistical Expert Systems. The earliest systems that provided advice on data analysis were statistical expert systems, which were based on rules defining when certain statistical techniques were useful or valid. REX [Gale 1986] covered linear regression, SPRINGEX [Raes 1992] handled bivariate, multivariate and non-parametric statistics and Statistical Navigator [Raes 1992] covered techniques such as multivariate causal analysis and classification. These systems asked the user multiple choice questions in order to trigger the expert rules. For instance, in REX the user was asked to choose between a fast result or a more accurate result based on a larger set of tests. Some other systems allowed the user to explore the rules directly to learn more about the techniques, e.g. using a textual search function (KENS [Hand 1987; 1990]) or a help function with hyperlinks (NONPAREIL [Hand 1990] and LMG [Hand 1990]). In all these systems, advice was limited to the hardcoded expert knowledge and a reduced set of problems. They did not provide direct guidance, but informed the users what possible (correct) decisions could be made at the next step of the analysis.

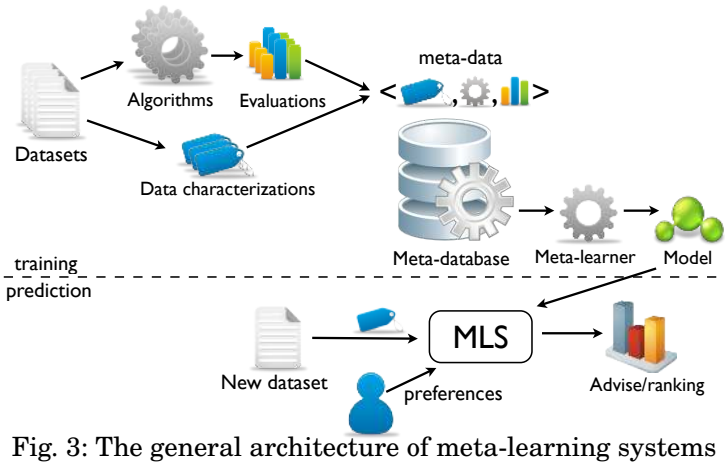

Fig. 3: The general architecture of meta-learning systems

Regarding their performance, we are not aware of any study which measures how good these systems were.

4.1.2. MLT Consultant. The first IDA for machine learning algorithms was Consultant-2 [Craw et al. 1992; Sleeman et al. 1995], which was linked to the Machine Learning Toolbox (MLT) [Graner et al. 1993; Kodratoff et al. 1992]. Its knowledge base stored about 250 heuristic expert rules. It interacted with the user through question-answering sessions, in which the user was asked to provide information about the data (e.g. the number of classes or whether it could be expected to be noisy) and the desired output (e.g. rules or a decision tree). Then the system utilized the stored rules to compute a score for each algorithm. After the user had selected an algorithm, the system ran this algorithm and afterwards engaged the user in a new question-answering session in order to assess whether the user was satisfied with the results. If not, the system would generate a list with possible parameter recommendations, which were again scored according to the stored heuristic rules.

The rules were based on user preferences and on important meta-features of the data (e.g. size, noisiness), the algorithms (e.g. relative memory usage, cpu time), and the produced models (e.g. the number of decision tree nodes or the average center distance of clusters). Further, these rules triggered actions such as adjusting algorithm scores and proposing parameter settings or data transformations [Sleeman et al. 1995].

4.2. Meta-learning systems

The previous systems were built with a fixed set of techniques in mind. However, as even more machine learning algorithms started to appear, it became increasingly important to find out which techniques were suitable for which kinds of data. This *algorithm selection problem* was formulated by Rice [Rice 1976] as the challenge of discovering the relationship between measurable features of the problem (the dataset) and the performance of different algorithms.

This can be seen as a standard learning problem: the relationship between the meta-data about the datasets and the algorithm performance could be *learned* (or modeled) and used to predict the best suitable algorithm for a given dataset. This led to the development of *meta-learning systems* (MLS), as shown in Figure 3.

Meta-learning systems consist of a training and a prediction phase. In the training phase, datasets are first characterized by a set of measurable characteristics. Many different characteristics have been defined including simple, statistical and/or

information-theoretical measures [Michie et al. 1994], concept-based properties [Vilalta and Drissi 2002a], model-based characteristics [Peng et al. 2002], landmarkers [Pfahringer et al. 2000] (evaluations of simplified algorithms), and subsampling landmarkers [Leite and Brazdil 2007] (evaluations of algorithms on several subsamples of the data). Subsequently, the algorithms are run on the dataset and their performance evaluations (e.g. error rate and speed) are linked to the characteristics of the involved dataset. In the training phase, a learning algorithm (called the meta-learner) is trained on all the collected meta-data, which will yield a model (e.g. a set of rules or a decision tree) of the algorithm performance on different types of data. Finally, the resulted model is used to make a prediction based on the characteristics of a new dataset. Most systems will return a ranking of the algorithms based on the predicted performance of each algorithm. Many also allow the user to define preferences, e.g. whether they care more about speed or accuracy.

4.2.1. StatLog. While some small meta-learning systems were proposed earlier, e.g., VBMS [Rendell et al. 1987], the StatLog project [Michie et al. 1994] was the first large-scale meta-learning study using 19 data properties and 10 algorithms. During the training phase, the system marked algorithms as *applicable* or non applicable, based on how close they were to the best algorithm on a given dataset. Then, a decision tree model was built for each algorithm predicting whether or not it is applicable on a new dataset. The end result was a set of learned rules that had to be checked manually.

4.2.2. The Data Mining Advisor (DMA). The Data Mining Advisor (DMA) [Giraud-Carrier 2005] further automated the idea from StatLog. It stored the actual performance measures for all algorithms and trained a k-Nearest Neighbor algorithm (k-NN) to predict how well the algorithms perform on a new dataset. It produced a ranking of all algorithms according to user-specified objectives. New datasets could be uploaded via a web-interface; its meta-features were automatically computed, and the ranking was returned subsequently. DMA used a set of 7 numerical data characteristics, 10 classification algorithms, and was initialized with 67 datasets, which originate mostly from the UCI repository[5]. The choice of a k-NN algorithm (with k=3) had the benefit that the meta-data from new algorithms or datasets could be added without retraining any models. All performance predictions were based on the three most similar datasets.

4.2.3. NOEMON. NOEMON [Kalousis and Hilario 2001; Kalousis and Theoharis 1999] follows the approach of Aha [Aha 1992] for comparing pairs of algorithms. Starting from a meta-database similar to the one used in DMA, the performance results on every combination of two algorithms are extracted. Then, the number of data meta-features is reduced using automatic feature selection. Here after, the performance values are replaced with statistical significance tests indicating in which situation one algorithm significantly outperforms the other. The results are then fed to a decision tree learner in order to build a model that predicts when one algorithm will be superior or when they will tie on a new dataset. Finally, all those pairwise models are stored in a knowledge base. At prediction time, the system collects all models relating to a certain algorithm and counts the number of predicted wins/ties/losses against all other algorithms to produce the final score for each algorithm. In the end, the final scores obtained in this manner serve for the rankings of the algorithms.

4.2.4. Other meta-learning systems. There are many more meta-learning algorithms described in the literature. Unfortunately, these fall outside the scope of this paper because they were never developed into IDAs. It is worth noting however, that many of them improve considerably upon the meta-learning systems described above. For

[5] `http://archive.ics.uci.edu/ml/`

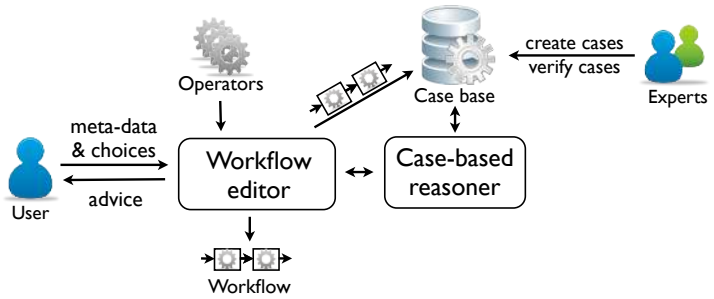

Fig. 4: The general architecture of case-based reasoning systems

instance, they introduce novel data characteristics such (subsampling) landmarkers, or employ different algorithms for building meta-models, such as boosted decision trees [Kalousis 2002], predictive clustering trees [Todorovski et al. 2002], regression algorithms [Bensusan and Kalousis 2001] and neural networks [Castiello and Fanelli 2005]. Some introduce new implementation frameworks, such as METALA [Botia et al. 2001; Hernansaez et al. 2004] and [Grabczewski and Jankowski 2007]. A very complete overview of these systems can be found in [Vanschoren 2010].

4.3. Case-based reasoning systems

Case-based reasoning (CBR) systems are related to meta-learning systems. Instead of learning a model during a separate training phase, they store all successful workflows, called *cases*, in a *case base*, which is maintained by (human) experts. Given a new problem, CBR systems provide advice based on these cases (see Figure 4).

The users input the new problem and their preferences into the system, which will return k previous workflows from the case base according to their similarity to the new problem and prior 'successfulness'. The latter is often defined by experts and/or feedback from previous users. In the next step, the user can select one of these cases and load it into a workflow editor, so that the case can be adapted to the new problem. The workflow editor allows to add or replace operators and will sometimes also provide additional guidance. After having finished this process, the new workflows can be uploaded as a new case to the case base. Usually, a group of experts is responsible for seeding the case base with useful cases as well as verifying new submissions.

Note that meta-learning systems that use a lazy learning algorithm as its meta-learner (e.g. k-NN) can rightly be conceived as CBRs, though, they store both successful and unsuccessful examples without expert involvement.

4.3.1. CITRUS. CITRUS [Engels 1996; Wirth et al. 1997] is built as an advisory component for Clementine [Grimmer 1996] – a well-known KDD suite, now part of IBM SPSS Modeler [6]. It contains a case-base of available 'processes' (KD operators) and 'streams' (sequences of processes), which were entered by experts and described with pre- and postconditions. In the beginning, the user has to provide an abstract task description, which is appended with simple data statistics. CITRUS will then perform CBR to load the most similar case in Clementine's main workflow editor, where it can be edited by the user. Additionally, CITRUS can check the validity of the user-edited workflows (using the stored pre- and postconditions), and facilitate the selection of operators by removing those operators that violate any constraints. Finally, it also uses an hierarchical planner to build partial plans, and decompose a task into subtasks for which smaller workflows can be defined.

[6] https://www-01.ibm.com/software/analytics/spss/products/modeler/

4.3.2. AST. AST [Lindner and Studer 1999] is a case-based reasoning variation of the Data Mining Advisor (DMA) approach, which also returns a single algorithm rather than a complete workflow. AST extends the numerical data characteristics with several simple, symbolic algorithm characteristics, such as whether the produced model is interpretable (true/false) and the relative training and testing time (fast/slow). The user is able to assign these preferences (e.g., 'the algorithm should be fast and produce interpretable models'), which are appended to the data characteristics calculated from the new dataset, thus forming a new case. Subsequently, AST selects an algorithm based on the three most similar cases and asks the user to adapt the workflow to the new problem. On the whole, the case base consists of 21 algorithms on 80 datasets with 16 StatLog data characteristics and 4 algorithm characteristics.

4.3.3. MiningMart. Likewise, the MiningMart project [Morik and Scholz 2004] aims at reusing successful workflows but primarily focusses on the data selection and data preprocessing phase (see Fig. 1). It is unique in that it has an elaborate workflow editor that allows the users to *map* a case directly to the SQL databases where the business data is stored, rather than expecting the user to first select and export the data in a format that the CBR system understands. MiningMart's online case base[7] stores workflows described in a specific XML-based language, which is named M4. The M4 language describes all details of the entire mapping process, as well as pre- and postconditions and characteristics of all operators, for being able to verify the validity of edited workflows. Instead of using a conventional CBR reasoner, Mining Mart describes all cases in an ontology with informal annotations, such as the goals and constraints of each problem. The user can then search the ontology and select a case. For mapping the case to the data stored in a database, it offers a three-tier graphical editor. First, in the *case editor*, the workflow can be adapted by adding or removing operators. All data in the workflow is described using abstract concepts and relations, such as 'customer', 'product', and the relation 'buys', which can each have a range of properties, such as 'name' and 'address'. Second, in the *concept editor*, these concepts can be edited to match the new problem (e.g., the property 'age' can be added to the customer). Also at this level, *concept operators* are defined, such as selecting or adding a property. Third, in the *relation editor*, these concepts, relations and properties have to be mapped to tables, columns, or sets of columns in the database. To run the workflow, MiningMart translates all steps into SQL queries and operator calls, and stores the preprocessed data in the database itself. At any point, the user can access these three editors for a further refinement of the workflow.

4.3.4. The Hybrid Data Mining Assistant (HDMA). The Hybrid Data Mining Assistant[8] (HDMA) [Charest et al. 2008] is a CBR system that offers additional guidance for the workflow editor based on expert rules (expressed in SWRL [Horrocks et al. 2004]), which are stored in an ontology. As such, it combines CBR with the expert rules of expert systems. In the case base, each case is described by 66 attributes: 30 attributes describing the problem (StatLog-like meta-features plus qualitative meta-features such as the type of business area, e.g. finance, or whether the data can be expected to contain outliers), 31 attributes describing the workflow (e.g. how outliers were handled or which model was used) and 5 'user satisfaction' ratings provided by previous users. All operators are likewise expressed in the ontology. The user first provides a dataset and the qualitative meta-features described above. The CBR system then returns two scores for each case: one based on similarity, the other based on user satisfaction. After a case has been selected, the system guides the user through this case in five distinct

[7]MiningMart's online case-base: http://mmart.cs.uni-dortmund.de/caseBase/index.html
[8]This is not the official name, we only use it here to facilitate our discussion.

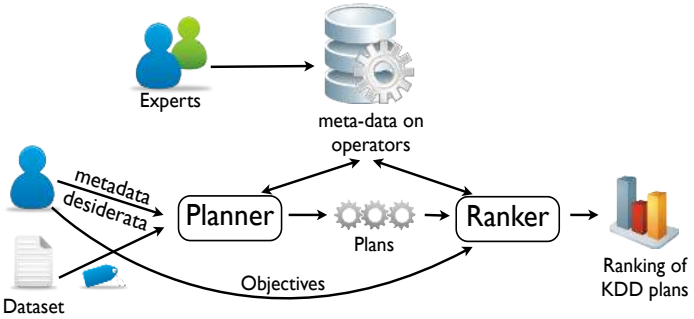

Fig. 5: The general architecture of Planning-based Data Analysis Systems

phases, corresponding to the KDD-process (see Fig. 1). In each phase, the user is given the solution of the selected case, as well as all operators for that phase. A rule reasoner further displays recommendations, which may encourage or advise against an operator based on the expert rules, e.g. "use an *aggregation* technique if the *example count* is greater than 30000 and the data is of a *transactional* nature", or "if you select the *Naive Bayes* technique, then a *nominal target* is required".

A similar approach to HDMA is the OLA assistant [Choinski and Chudziak 2009] which employs ontologies combined with CBR focusing mainly on the collaboration between domain and technology experts. For this purpose both domain and DM ontologies are used, the later even aligned along the CRISP-DM model. Opposed to other CBR approaches OLA is not limited to the KDD process but it follows the CRISP-DM phases. It also uses the rules stored in ontologies and IOPE to dynamically compose, rank and present to the user valid processes. Unfortunately there are many missing details about this approach. There is no public knowledge if it was implemented/finished, therefore we did not add it in our comparison section.

4.4. Planning-based Data Analysis Systems

The previously discussed systems share a common weakness: if the given problem is different from the ones seen before, the IDA cannot provide other support than returning general expert advice. In these cases, it becomes crucial that the IDA is able to construct and evaluate workflows autonomously, so that it can offer good candidate solutions to the user. One way to do this is to view the workflow construction problem as a *planning* problem, in which a plan must be built consisting of operators that transform the initial data into accurate models or predictions.

The general architecture of these Planning-based Data Analysis Systems (PDAS) is shown in Figure 5. First, additional meta-data about the involved operators needs to be stored. This includes not only the inputs, outputs, preconditions and effects (IOPE) of each operator but can also comprise other meta-information that we have encountered before, e.g. the operator's relative speed, to evaluate the resulting plans. First, a planning domain description needs to be built (e.g. in PDDL [McDermott et al. 1998]) based on the dataset characteristics and additional information from the user. Next, the planning component will generate valid KD workflows (plans). Finally, these workflows are ranked according to the user's objectives.

4.4.1. AIDE. AIDE [Amant and Cohen 1998a] is a mixed-initiative planning system designed to help users during exploratory data analysis. In mixed-initiative systems, the system starts a dialogue with the user: it gives recommendations about which operators to use next and in turn allows the user to review and override these recom-

mendations. For planning, it uses an approach similar to Hierarchical Task Network (HTN) planning [Erol 1996], in which the dependencies between actions can be given in the form of networks. This allows to plan on a higher, more abstract level, and to decompose a planning problem into smaller sub-problems with subgoals. The system thus builds high-level plans consisting of sub-problems as well as low-level plans in which each subproblem is solved by workflows of primitive operators. Moreover, AIDE also resembles CBR systems: it has a plan library (a case base) which is used to extract plans for similar cases. These plans can also consist of subproblems rather than primitive operators. Its planner is script-based and offers 3 primitive operations: reduction, transformation, and decomposition, which are used for data manipulation in the exploratory process.

4.4.2. The Intelligent Discovery Electronic Assistant (IDEA). A first PDA system for DM is the Intelligent Discovery Electronic Assistant (IDEA) [Bernstein et al. 2005]. It regards preprocessing, modeling, and post-processing techniques as operators and returns *all* valid plans for the given problem. The IOPE information is encoded in an ontology, which also contains manually defined heuristics (e.g., the relative speed of an algorithm). Before the planning process, the user is asked to give weights to a number of heuristic functions, such as model comprehensibility, accuracy, and speed. After planning, the workflows are passed on to a heuristic ranker, which uses the heuristics enlisted in the ontology to compute a score congruent with the user's objectives (e.g., building a decision tree as fast as possible). Finally, based on this ranking the user may select a number of processes to be executed on the provided data. After the execution of a plan, the user is allowed to review the results and alter the weights to obtain alternative rankings. For instance, the user might sacrifice some speed in order to obtain a more accurate model. Finally, if useful partial workflows have been discovered, the system also allows to extend the ontology by adding them as new operators.

4.4.3. NExT. NExT, the Next generation Experiment Toolbox [Bernstein and Daenzer 2007], is a CBR-extension of the IDEA approach. It contains a case base of past workflows and uses CBR to retrieve the most useful ones. Often, only parts of these workflows will be useful, leaving gaps that need to be filled with other operators. For instance, a workflow may start with a preprocessor that cannot be applied on the given data. This is where the planning component comes in: using the preconditions and effects of all operators it will try to find new sequences of operators to fill in those gaps. NExT includes an ontology of possible problems related to workflow execution and matching resolution strategies. Calling a planner is one such strategy, other strategies may entail removing operators, or even alerting the user to fix the problem manually. During operation, NExT records a complete log of experimentation, which can be shared with other researchers interested in reproducing the workflow. Although evaluations of the NExT system remain scarce, its reuse of prior workflows and semi-automatic adaption of these workflows to new problems seems promising.

4.4.4. KDDVM. The Knowledge Discovery in Databases Virtual Mart (KDDVM) system [Diamantini et al. 2009b] is based on the KDDONTO ontology, which likewise describes the algorithms with their inputs and outputs as well as their properties and relationships. Akin to RDM, KDDVM interacts with the ontology by using the Pellet reasoner and SPARQL-DL queries. Instead of applying a standard planning algorithm though, it utilizes a custom algorithm that starts at the goal state and iteratively adds operators, thus forming a directed graph until the first operator is compatible with the given dataset. Operators are added using an *algorithm matching* procedure, which checks the compatibility of inputs and outputs. However, the operator interfaces do not need to be perfectly equivalent: their *similarity* is computed through their dis-

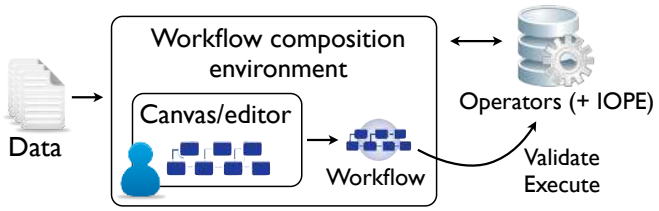

Fig. 6: The architecture of workflow composition environments

tance in the ontology graph. Finally, the produced workflows are ranked based on the similarity scores as well as other operator properties stored in the ontology (e.g. soft preconditions or computational complexity estimates).

4.4.5. RDM. The RDM system [Žáková et al. 2010] uses an OWL-DL-based [Patel-Schneider et al. 2004] ontology for knowledge discovery. It is organized around the concepts of knowledge (datasets, constraints, background knowledge, rules, etc.), algorithms (inductive algorithms and data transformations), and the knowledge discovery task. This ontology is queried using the Pellet reasoner [Sirin et al. 2007] and SPARQL-DL queries [Sirin and Parsia 2007], for retrieving the operator inputs and outputs, which are then fed into the planner. Two planners are implemented. The first one is a standard PDDL planner, thus the inputs and outputs are first translated into PDDL before the actual planning phase. The second is a modification of the heuristic Fast-Forward (FF) planning system [Hoffmann and Nebel 2001]. Here, the ontology is queried directly during the planning process. Additionally, it uses a heuristic to guide the planning: given the current partial plan, the final length of the plan is estimated using a secondary planner (GRAPHPLAN [Blum and Furst 1997]).

4.4.6. eIDA. The eLico-IDA (eIDA) is a product of the eLico project[9]. It deploys the Data Mining Workflow Ontology (DMWF) [Kietz et al. 2009], which stores not only operator inputs and outputs but also preconditions and effects in the form of SWRL rules (stored as annotations in the ontology). eIDA uses a hierarchical task network (HTN) planner implemented in Flora2 [Yang et al. 2002]. The ontology is queried and all inputs, outputs, preconditions and effects are translated to Flora2 for planning. The resulting plans are then ranked using a second ontology, DMOP [Hilario et al. 2009], which stores detailed properties of the operators. The system also offers a modeling tool, eProPlan, for defining the planning domain and user preferences (e.g. the DM task). Moreover, it provides an API interface to DM suites and can be used as a plugin for RapidMiner 5 [Mierswa et al. 2006] and Taverna [Oinn et al. 2004].

4.5. Workflow Composition Environments

The final category of systems do not automatically propose operators or workflows, but rather support the user during manual workflow composition by offering a graphical environment, where the data flow can be drawn on a canvas, or a high-level scripting language for quick workflow design and execution (see Figure 6).

WCEs are the baseline of this survey since they are the only ones actively used, mainly because they focus on ease-of-use and actually executing the workflow. They are not only a collection of algorithms, but they also offer some guidance (e.g., auto-wiring, meta-data propagation, correctness checking before execution, operator recommendations, etc.) and therefore can be considered 'pseudo-IDAs'.

[9] http://www.e-lico.eu/

4.5.1. Canvas-based tools. IBM SPSS Modeler[10] is a commercial DM tool that allows the user to easily design DM workflows via an intuitive graphical interface. It contains many DM operators and provides support for all KDD steps. It also incorporates the statistics capabilities of SPSS-like data transformation, hypothesis testing, and reporting capabilities. In addition, makes use of advanced analytical functions, automated data preparation, and rich, interactive visualization capabilities.

SAS Enterprise Miner [Cerrito 2007] uses the SAS methodology, SEMMA [11], designed to handle large data sets in preparation for subsequent data analysis. It provides extensive support for statistics, numerous types of charts and plots, and data exploration via hierarchical market baskets and multivariate graphical data exploration. A new feature is the SAS Rapid Predictive Modeler which offers the user an automatic guidance through the data preparation and all other DM tasks.

Weka [Hall et al. 2009] is the first open-source suite of machine learning algorithms for DM tasks. It provides support for several KDD steps like preprocessing, feature selection, DM step, evaluation, and visualization of results. Moreover, people have developed a various packages for additional tasks (e.g., tree visualization). Recently, the suite has even been integrated in the newer DM tools, like RapidMiner, KNIME, etc.

RapidMiner [Mierswa et al. 2006] is the most popular open-source system for DM.[12] It has both a GUI mode and a command-line server mode. It can be easily extended and offers many available extensions. It encompasses more than 500 operators for data integration and transformation, DM, evaluation, and visualization tasks. Moreover, it provides powerful high-dimensional plotting facilities.

KNIME [Berthold et al. 2009] is a younger system that enables the user to easily design, execute, and investigate DM workflows as well as integrate new DM algorithms. It incorporates over 100 operators for reading and exporting data (input/output), preprocessing and cleaning, modeling, analysis and DM as well as several interactive data visualizations. KNIME is compatible with Weka and the R framework via plugins.

4.5.2. Scripting-based tools. Another important category are the script-based mathematical tools which originally did not focus on DM but rather on mathematical and visualization functions that also allow to implement DM algorithms.

MATLAB [MathWorks 2004] is a high-level language and framework that enables users to quickly develop algorithms, analyze data, and visualize it. Its language can also be used to define workflows as small programs. This tool is mainly used for all sorts of basic data analysis, like basic statistics, matrix processing and curve fitting. Additionally it contains many extensions to more advanced analysis techniques, such as computer vision. It further supports a couple of graphical interfaces, which simplify the interaction with the toolbox. For example, Gait-CAD [13] is designed for the visualization and analysis of time series and features with a special focus on DM problems including classification, regression, and clustering.

R [Ihaka and Gentleman 1996] is a framework for data analysis based on the high level language R and used to provide statistical functionality. As such, it resembles MATLAB, but instead focuses on data analysis: it has various statistical and graphical techniques (e.g., linear and nonlinear modeling, classical statistical tests,etc.). Many scientists prefer it because it is very extensible: new algorithms and techniques can be easily added. Further, various GUIs have been developed for it from which the

[10] http://www-01.ibm.com/software/analytics/spss/products/modeler/
[11] http://www.sas.com/offices/europe/uk/technologies/analytics/datamining/miner/semma.html
[12] http://www.kdnuggets.com/polls/2011/tools-analytics-data-mining.html
[13] http://www.iai.fzk.de/www-extern/index.php?id=656&L=1

most known is Rattle [14]. Rattle features several windows/tabs for visualizing data and results. Another GUI for R that supports interactive graphics is RStudio [15].

5. COMPARISON

This section presents a comparison of various IDAs from a new perspective by considering both the background knowledge they use and the support they provide. The purpose of this comparison is to organize the existing IDAs along a set of dimensions, to identify trends as well as their assets and drawbacks.

5.1. Background knowledge

First, we compare all IDAs based on their prior knowledge of data analysis techniques (see Section 3), or lack thereof. This will give each system certain advantages or disadvantages compared to other IDAs. The results are summarized in Table I. The first column states the IDA category, as established in the previous section: expert systems (ES), meta-learning systems (MLS), case-based reasoning systems (CBR), planning-based data analysis systems (PDAS) and workflow composition environments (WCE). The next column depicts which of the five phases of the KDD-process (see Figure 1) are covered by the system: data selection, preprocessing, transformation, modeling/mining and evaluation, in that order. The remaining columns cover, from left to right, which meta-data the IDAs extract from the input data, which meta-data they hold about the operators, whether they use models (e.g. rules) predicting which operators to use, and whether they use a case base of prior, successful cases. Finally, the last column briefly states the type of advice each system offers to the user. In the remainder of this section, we will explain how each IDA uses these types of knowledge in order to offer specific types of support to the user, and perhaps more importantly, which types of knowledge are missing in IDAs that in turn could be used to improve them.

5.1.1. Range of operators. The ideal IDA should know about all available data analysis operators. Usually, however, this is a limiting factor for many of the discussed IDAs. ES and MLS systems, for instance, control only data mining operators, and thus can only provide advice on which data mining technique to select. As such, they are only useful in cases where the data is already preprocessed considerably by the user, and the only remaining steps are modeling the data and evaluating the resulting models. Several systems, such as DMA, and the ES systems can automatically evaluate their own suggestions and update their advice based on the outcome.

CBR, PDAS and WCE, also control preprocessing operators (e.g., missing value imputation) as well as transformation operators (e.g., feature selection). As such, they can provide multi-step support starting from an input dataset: given some pre-selected, partially cleaned data, they can provide advice on which series of techniques are best used. However, few cover the initial data selection step. The only exceptions are MiningMart, which offers a mapping interface to map workflows directly to the raw data stored in relational databases, and WCE systems, which do not help the user with deciding which data to select, but do offer operators for extracting the data from databases. In short, these systems are much more useful when starting with raw data.

There is also a difference in depth: most IDAs only support a (few) dozen techniques in each KDD phase, and are not always easily extensible (e.g. new expert rules or cases are needed). This may limit their usefulness in practical scenarios. For instance, if no operators are included for image processing, they won't be able to provide useful advice for image data. WCE typically control hundreds of operators, covering a wide range of

[14] http://rattle.togaware.com/
[15] http://rstudio.org/

Category	System	Range of operators	Meta-data on input dataset	Meta-data on operators		Predictive models	Case base	Type of advice
				External (IOPE)	Internal			
ES	Statistical expert systems		simple data properties	–	–	hardcoded expert rules	–	advice on techniques via Q&A sessions
	MLT Consultant		simple data properties	–	speed, model complexity, ...	250 heuristic expert rules	–	scores ML methods + parameter settings
MLS	StatLog		19 measurable data properties	–	operator evaluations	decision tree per ML algorithm	–	returns all "useful" algorithms (defaults)
	DMA		7 measurable data properties	–	operator evaluations	k-Nearest Neighbor	–	returns ranking of 10 algorithms (defaults)
	NOEMON		35 measurable data properties	–	operator evaluations	decision tree per algorithm pair	–	returns ranking of 3 algorithms (defaults)
CBR	CITRUS		simple data props + info from user	pre- & postconditions (hardcoded)	–	–	partial workflows	finds similar flows + checks validity
	AST		16 measurable data properties	–	speed, model interpretability	–	21 single operators	selects best from 21 algorithms (defaults)
	MiningMart		info from user	pre- & postconditions (M4 format)	–	–	workflows + mappings	remapping of flows + checks validity
	HDMA		30 data properties + info from user	pre- & postconditions	–	expert rules in SWRL	workflows, 36 properties	ranks cases + helps user adapt them
PDAS	AIDE		info from user	operator (sub)goals	–	–	(abstract) workflows	mixed initiative: proposes useful steps
	IDEA		simple data props provided by user	IOPE's (STRIPS, in ontology)	heuristics (e.g. relative speed)	–	–	returns ranking of all valid workflows
	NeXT		simple data props provided by user	IOPE's (STRIPS, in ontology)	–	–	prior workflows	finds similar cases + fixes workflows
	KDDVM		general task input description (user)	IOPE's (in ontology)	algo properties in ontology	–	–	builds approximate and exact workflows
	eIDA		task description (user+computed)	IOPE's (SWRL, in ontology)	algo properties in ontology	–	–	generates and ranks workflows
	RDM		general task input description (GUI)	IOPE's (in ontology)	algo properties in ontology	–	–	generates abstract & specific workflows
WCE	Script-based		at operator level	inputs & outputs (hardcoded)	–	–	–	powerful high-level scripting language
	Canvas-based		at operator level	inputs & outputs (hardcoded)	–	–	–	visualize inputs, outputs,connections

Table I: Overview of IDAs by their background knowledge

data analysis tasks, but unfortunately provide only limited advice. PDAs are generally more easily extensible, and are sometimes built on top of WCEs (e.g. eIDA uses RapidMiner), which makes them the most widely applicable IDAs. Extending them typically means adding the operators and their IOPE information in their ontologies.

5.1.2. Meta-data on input dataset. Ideally, IDAs should be able to automatically extract all useful information from the given input data, so that it can give targeted advice on which techniques are most useful in each specific situation. Unfortunately, each IDA uses only a specific subset of that information, which has a profound effect on the advice it is able to offer. ES systems use rather simple measures, such as the number of data points or the number of attributes, which are indeed useful but not indicative for subtle differences between operators. MLS systems, on the other hand, use a vast array of measurable data features that affect the operator performance. These include statistical (e.g. skewness) and information-theoretic (e.g. entropy) measures [Michie et al. 1994; Castiello et al. 2005], which are applied in StatLog, DMA and NOEMON. Meta-learning systems use this meta-data to select useful operators, while CBR systems use similar measures for calculating the similarity between a new case and prior ones from the case base.

Interestingly, more successes were recorded for model-based measures [Peng et al. 2002], e.g. make use of the complexity of a decision tree trained on the data to characterize this data, and landmarking [Pfahringer et al. 2000], i.e. use the performance of fast simplified algorithms (e.g. a decision stump learner) to characterize the data.

However, these model-based measures were never included in an IDA. Moreover, most meta-learning research has focused on tabular data. For many other types of data (e.g., graphs or text) there exist only limited work on identifying which data properties would be good indicators of operator suitability. None of the presented IDAs are able to provide targeted advice in such situations, especially not the ES, MLS and CBR systems, which are very dependent on good input meta-data to find similar cases or build accurate predictive models. As such, an important direction for future work will be to include more detailed input data properties on a larger variety of tasks.

PDAS systems often depend on interaction with the user to obtain information on the given task. For instance, if an operator only works on nominal data, then the PDAS will either measure this property in the input data (as in eIDA), or, more often, ask the user to manually fill in such properties as part of the planning task description. In addition, the user is supposed to indicate the desired end state of the planning process, which is typically defined as a *task*. For instance, in 'predictive modeling' tasks, the user expects a predictive model at the end of the workflow. In eIDA, which employs an HTN planner, the user is even able to define subtasks, such as attribute selection or discretization, which need to be fulfilled along the way. As such, they are more widely applicable, but also require more user expertise in defining the problem. Finally, note that while a PDAS will be able to build workflows as long as the necessary IOPE information is available, the workflows may be of lesser quality if there is less guidance (e.g., in the form of heuristics) on which operators to select for which types of data. We will discuss this in the next section.

5.1.3. Meta-data on operators: IOPE. Next, IDAs should be able to link information on the input data to the properties of all operators (or entire workflows), in order to generate only the most useful workflows.

Table I shows that all IDAs use at least some meta-data on both the input data and operators. As said before, we distinguish between *external* (black-box) properties, i.e. the operator's inputs, outputs, preprocessors, and effects (IOPE), and *internal* properties, which are linked to the operator's structure (e.g. model type) or performance (e.g. speed, accuracy and model complexity).

IOPE information allows CBR, PDAS and WCE systems to verify the validity of workflows (e.g., no preconditions are violated). More importantly though, IOPE can be used to describe a planning domain and leverage AI planning to automatically generate valid workflows. This information is either hardcoded (CITRUS and WCE systems), encoded in standard formats (e.g., IDEA and NEXT use the STRIPS format), a custom format (MiningMart) or stored in ontologies. eIDA expresses IOPE information in SWRL rules, which are stored in an ontology as annotations, and AIDE stores IOPE information as operator (sub)goals to drive its HTN planning.

5.1.4. Meta-data on operators: internal properties. Nevertheless, generating all valid workflows is not very helpful if there exist hundreds of them. Additional information is needed for being able to rank the workflows according to user preferences or the properties of the input data.

The simplest solution is to rank workflows according to their complexity (e.g. number of operators), as in the RDM system. MLT Consultant, AST and IDEA, on the other hand, make use of relative performance metrics, such as the relative speed, memory usage, and model complexity. For instance, if a user prefers a fast solution, then the speed of a workflow is estimated based on the speed of the constituent operators and scored accordingly. RDM, KDDVM and eIDA store similar properties in their ontologies (e.g., computational complexity indices), in addition to task-oriented properties, such as whether a classification model is probabilistic or not. This is very useful when the user has specific requirements for the workflows.

A problem with such operator performance properties is the fact that they are obtained via averaging the operator's performance over different applications, and hence may not be accurate for the given input data. MLS systems, therefore, store all operator evaluations in a separate database and link them to the properties of the input data. This is done by training a predictive model on the meta-data of the input data and the operator's performance evaluations. As such, these models can predict the operator performance much more accurately given precise properties of the input data.

5.1.5. Predictive models. An ideal IDA should also be able to *learn* from previous tasks, and use that experience to perform future tasks better. ES systems only use hard-coded rules handcrafted by human experts. In MLT Consultant, these rules are used to update the scores for each DM technique, or generate scores for specific parameter settings. Support for parameter optimization is a quite unique feature. WCEs only support exhaustive or manual parameter optimization, and most other systems do not consider parameters at all.[16]

The problem with expert rules is that for every new operator, new rules have to be defined that differentiate the new operator from all the existing operators. Given that new techniques are introduced at a constant rate, this quickly becomes infeasible. MLS systems solve this issue by automatically training predictive models on meta-data. As such, their models are automatically updated to cover new operators, and predictions generally improve as more examples of data analysis problems are encountered.

However, all predictive models used so far only predict a single operator – almost always a DM operator, but never entire workflows. This is problematic because choosing the right data preprocessing and transformation techniques typically has a much greater effect than selecting the best DM technique. HDMA, on the other hand, takes a middle course by first selecting a prior workflow and then using expert rules to adapt that workflow to the new data. Unfortunately, the rules are merely used to inform the user, rather than automating the workflow composition process.

Predictive models are never used in PDAS or WCE systems,[17] which is unfortunate. It seems very useful to use predictive models to predict the effects of each operator given the input data it receives. As such, predictive models could be used to predict which operators are most likely to perform well at a certain point in the workflow. In heuristic planners, these predictions can be part of the heuristic. Furthermore, after executing an automatically generated workflow, it seems perfectly sensible to calculate the actual data characteristics for each point in the workflow and use predictive models to predict which operators might be even better based on prior observations. This seems to be a promising avenue for future work.

5.1.6. Case base. Case bases form an important alternative to predictive models: instead of predicting how well an operator or workflow will perform, they only store successful workflows, assuming those will always perform well if the input data is similar. The latter is estimated based on measurable properties of the input data: if a prior case was related to data with similar properties, then the associated workflow will likely perform well. As such, all systems that use a case base must also calculate a range of meta-data on the input dataset. Especially HDMA uses a quite large array of data properties, which improves the similarity calculations.

[16] Some meta-learning studies were performed that treated different parameter settings as different algorithms [Soares et al. 2004], but no meta-learning system has been developed that provides accurate advice on both algorithms and parameter settings.

[17] eIDA has SWRL rules, but they are not predictive: they only check the validity of an operator in a workflow.

Like systems that use predictive models, CBR systems also automatically improve as more people use it, since this will enrich the case base. The best-practice cases developed by experienced users are stored for later re-use, which saves time and costs.

However, CBR systems still rely heavily on experts to seed and maintain the case base: it needs to be densely populated and unsuccessful workflows should be avoided. Indeed, if no sufficiently similar workflow exists in the workflow, they will not be able to offer much assistance. Moreover, after a similar prior case has been found, the remaining problem *adapts* the associated workflow, so that it will work on the new data. On this aspect, the different systems vary greatly.

CITRUS and AIDE return only partial or abstract workflows, leaving it to the user to complete the workflow. MiningMart offers much greater support: it uses various graphical editors to (manually) map the prior workflow to the new data: additional operators can be added, data attributes (*concepts*) can be added or removed, and data attributes can be *mapped* to (sets of) columns in a relational database. As such, it will be especially useful for situations in which data is stored in relational databases. Moreover, MiningMart includes *cases with self-adapting operators* by using multi-strategy learning [Kietz et al. 2000], which may greatly help novice users. It also provides a nice user interface with facilities for storing and retrieving cases.

HDMA uses a different approach entirely: it applies expert rules to advise the user about useful changes. For instance, if the prior case did not have missing values but the current one does, then the system will advise the user to add an operator that imputes missing values, because it has an expert rule that says so. This interesting mix of expert rules and CBR provides more targeted guidance for novice users. However, it also entails that, for every new operator, new rules have to be entered by the experts. As such, it will only perform well if both the case base and expert rule base are continuously maintained by experts. A more important problem, however, is that the utility of a case may diminish in later phases of the KDD-process: if the user chooses a different preprocessing step, then the subsequent stages of this case (e.g. data modeling steps) may lose their applicability, thus reducing the quality of the final workflow.

NeXT offers perhaps the most complete solution for novice users: it first retrieves the most similar workflow from a case base, and if the workflow fails on the new data, the system tries to automatically fix the workflow by using planning to complete the 'gaps' in the workflow or by removing unnecessary operators. This is an interesting combination of case bases and IOPE meta-data, although evaluations of the system remain scarce.

DMA is related to CBR systems: it uses a Nearest Neighbor classifier which stores evaluations of induction algorithms and measures dataset similarity to predict the best algorithm for a new dataset [Hilario and Kalousis 2001]. However, it does not support multi-operator workflows. Finally, AIDE stores previous abstract workflows which it can match with new workflows to reuse previous results and decisions, though it is not actually a CBR system.

5.1.7. Final remarks. From this overview, it is clear that no system fully exploits all available types of background knowledge. ES and MLS do not use IOPE information and therefore are not able to construct entire workflows. Conversely, PDAS generate and rank workflows based on user preferences but in essence they start from scratch every time. They do not capitalize on experience gained from previous data analyses. If they stored performance evaluations of the operators in the generated workflows, they could make use of that information during planning in order to get better estimates of the usefulness of the operators and generate more appropriate workflows. The vast array of input data properties used in MLS and CBR systems could also be used to improve the planning process. For instance, if a data mining operator is known to

perform badly on skewed data distributions, it makes sense to store this property in the ontology as an operator property and measure the data skewness in the input data.

Regarding how the different IDAs influenced each other, it is clear that early statistical expert systems influenced MLT Consultant. Further, StatLog influenced DMA, NOEMON and AST, and IDEA influenced RDM, KDDVM, eIDA and NeXT. However, there is almost no cross-fertilization between different categories of IDAs. The only exceptions are HDMA, which combines CBR with expert rules, and NeXT, which combines CBR with planning. Indeed, there seems to be a lot of unexploited potential in trying to combine these different approaches.

Finally, with the exception of RDM, KDDVM and eIDA, all IDAs use their own custom sets of operators. This generally means that they offer a lot of guidance on a very restricted set of operators. WCEs, on the other hand, offer very limited guidance on huge sets of operators. It seems to be clear that future IDAs are best developed on top of such WCEs – a trend that is especially clear in the newest IDAs: RDM, KDDVM and eIDA. Challenges remain however, as some approaches do not scale well to hundreds of operators: MLS systems require much larger repositories of operator evaluations for training models on them and CBR systems require much larger case bases. For planning-based systems, it quickly becomes infeasible to generate all valid plans, thus generating a necessity for finding solutions to generate only the best plans.

5.2. Type of support

Second, we compare the IDAs based on the support they provide for facilitating the data analysis process, as described in Section 2. Table II summarizes the findings on the kind of support the systems offer, their status, and availability as well as relevant references for each of them in the last columns. The user support can be divided into two main categories: first, the workflow-based support, like the first six entries of the table; second, the user-based support. In our discussion we group the types of support which have contradictory/opposite purpose (e.g., single step vs. multi-step, automatic generation vs. manual, etc) in order to better emphasize their contributions.

5.2.1. Modeling a single-step vs. multi-step from the KDD-process. A first distinction of IDAs is based on the complexity of the DM advice offered. Here we distinguish between systems which offer advice on one step of the KDD-process and the ones which recommend all of them. An ideal IDA should include both types: the first helps to decide when to use a certain operator and the second what operators should be used in a sequence.

For instance, ES provide support for a single step of the KDD-process. The advice is based on the information provided by the user combined with the expert rules from the knowledge base. Most often the advice is guided by questions addressed to the user and concerns both the modeling and the evaluation steps. Consultant-2 is the most advanced ES system. It suggests new methods in case the one applied has produced unsatisfactory results. Thus, a step from the KDD-process is not seen as a single-step but as a cyclic process, where the user can reapply other algorithms in case the current results are not satisfactory. MLS systems have a similar focus: they recommend the most appropriate algorithm for a certain data set – actually, they focus primarily on the modeling step (classification and regression).

Nevertheless, optimizing a single step is not an easy task. If we observe the modeling step, for instance, it becomes apparent that each algorithm has several parameters. For a naive user with little understanding of the DM domain, it is not trivial to set their values. Probably most of these users prefer to apply just the default values.

The shift from single-step to multi-step was initiated by the CRISP-DM standard and the CITRUS system. Both approaches guide the user through all phases of the KDD-process. WCE, on the other hand, are neutral, since they allow the users to ex-

Category	System name	KDD single step support	KDD multi-step support	Graphical workflow editing	Automatic workflow generation	Workflow checking/repair	Re-use past experiences	Task decomposition	Design support	Explanations	Experimental	Analytical	DM experience level	Status/Availability	References
ES	SES	++	−	−−	−−	−−	−−	−−	−−	++	−	++	naive (REX), experienced	NA/O	[Gale 1986], [Raes 1992], [Hand 1987; 1990]
	MLT Consultant	++	−	−−	−−	−−	−−	−−	−−	++	++	++	all	NA/O	[Sleeman et al. 1995]
MLS	StatLog	++	−	−	−	−−	−	−	−	−	−	++	unspecified	-/O	[Michie et al. 1994]
	DMA	++	−	−	−	−−	−	−	−	−	+	++	experienced	A/O	[Giraud-Carrier 2005]
	NOEMON	+	−	−−	−−	−−	−−	−−	−	−	+	+	unspecified	NA/O	[Kalousis and Hilario 2001]
CBR	CITRUS	−	++	−	+	−−	++	++	−	++	+	−	all	NA/O	[Engels et al. 1997]
	AST	+	−	−	−−	−−	++	−−	−	−	+	−	unspecified	NA/-	[Lindner and Studer 1999]
	MiningMart	0	0	+	−	−−	++	−	−	−	++	−	experienced	A/O	[Morik and Scholz 2004]
	HDMA	0	0	−	−	−−	++	−	−	+	+	−	all	NA/U	[Charest et al. 2008]
PDAS	IDEA	−	++	−	++	−−	−	−	−	−	+	−	intermediary	NA/O	[Bernstein et al. 2005]
	RDM	−	++	−	++	−−	−	−	−	−	+	−	intermediary	NA/U	[Žáková et al. 2010]
	KDDVM	−	++	−	+	−−	−	−	−	−	+	−	intermediary	A/U	[Diamantini et al. 2009b]
	eIDA	+	++	+	++	++	+	++	+	−	+	−	intermediary	A/U	[Kietz et al. 2010]
	NeXT	0	+	+	+	−−	+	−	+	−	+	−	intermediary	NA/O	[Bernstein and Daenzer 2007]
	AIDE	+	+	−	+	−−	+	+	−	+	+	−	intermediary	NA/O	[Amant and Cohen 1998b]
WCE	IBM SPSS Statistics	0	0	−	−	−−	−	−	−	+	++	−	experienced	A/U	[Levesque 2005]
	R	0	0	−	−	−	−−	−	−	−	++	−	experienced	A/U	[Ihaka and Gentleman 1996]
	MATLAB	0	0	−	−	−−	−	−	−	−	++	−	experienced	A/U	[MathWorks 2004]
	IBM SPSS Modeler	0	0	++	−	++	−	−	++	++	++	−	experienced	A/U	Link to project site[1]
	SAS Enterprise Miner	0	0	++	−	−−	−	−	++	++	++	−	experienced	A/U	[Cerrito 2007]
	Weka	0	0	++	−	−−	−	−	+	+	++	−	experienced	A/U	[Hall et al. 2009]
	RapidMiner 5.0	0	0	++	−	++	−	−	++	+	++	−	experienced	A/U	[Mierswa et al. 2006]
	KNIME	0	0	++	−	−−	−	−	++	+	++	−	experienced	A/U	[Berthold et al. 2009]

[1] `http://www-01.ibm.com/software/analytics/spss/products/modeler/`

++	= well supported (a main feature of the system)		A	=	publicly available
+	= supported		NA	=	not publicly available
0	= neutral, the system can do it but there is no assistance		U	=	up-to-date
−	= not present but integrable		O	=	outdated
−−	= not possible				

Table II: Overview of IDAs by offered support

ecute single operators for which, however most often, they have to set the parameters themselves. They further enable the user to design and execute multi-step KDD-processes, but this becomes hard if the processes have a large number of operators.

Above all, PDAS and partially CBRs support multi-step recommendation. They recommend correct workflows by taking the operators' preconditions and effects into account. Also some of the parameters are set (e.g., size of bins, number of bins). IDEA [Bernstein et al. 2005] provides its users with systematic enumerations of valid KDD-processes and rankings by different criteria. However, the system does not support the user through the steps of the KDD-process since it merely enumerates all steps. The workflow composition involves choosing tinduction algorithm(s) and appropriate pre- and post-processing modules. Though, it is limited to only a few operators for

each step. Opposed to IDEA, AIDE [Amant and Cohen 1998b] does not provide support for the overall KDD-process. For example, if you load a data set, the steps that are applied are: collecting information about each variable, the generation of statistical summaries, testing of the data types, testing of the continuous or discrete data, generation of hierarchical clusters for numerical data. Then a set of indications are generated, like indication of skew or indication of clustering tests for outliers. Further on, the user can make decisions and influence the execution of the plan generator.

In conclusion, we observe that most of the user support concerns the modeling steps or the automatic generation of all steps. However, preprocessing and feature selection steps are rarely addressed.

5.2.2. Graphical editing vs. automatic generation. Here we compare systems which are able to automatically generate workflows, i.e., PDAS, to systems which force the user to design the workflows manually. Ideally, these aspects should be both integrated into IDAs, automatic generation should be on demand and the graphical editing as default.

IDEA uses straightforward search for automatically outputting the valid processes. Here the user can select the plan by choosing a ranking method (accuracy, speed, etc.). The approach in AIDE differs from IDEA in the manner in which the user and the machine interact: AIDE offers a step by step guidance based on the script planner and the user's decisions. This is suitable for exploratory statistics but is not suitable for domains where algorithms run for an extensive period of time. Similarly, CITRUS combines planning with user guidance in order to help the user construct the best plan using a limited number of operations. IDEA only enumerates workflows, but it does not allow the user to modify neither nodes nor parameters. However, AIDE is user-centric and can support modifications at any decision point. The user is even able to go back several steps and modify decisions taken by the system. Beyond that, the workflows generated by IDEA are valid since the operators are modeled in an ontology with IOPE which need to be satisfied in order to be able to apply the operator. Moreover, IDEA provides an auto-experimentation module, which can execute the generated plans and rank them by accuracy.

Similar approaches are RDM and KDDVM, which generate abstract workflows using an ontology combined with planning and reasoning. RDM is able to generate several plans, however, there is no statement about what the maximum number could be. Žáková et al. [2010] exemplify one workflow for each of the application domains. In the evaluation section, the planner performance results are presented and four workflows are considered for each of the domains. Beyond that, RDM uses a repository for storing all the constructed workflows. When the user solves a DM task, the repository is searched in order to retrieve a solution, however only one. If no solution is found, the planner is called and several workflows are produced. The evaluation of the planner has been conducted on two application domains, namely CAD and gene data using two ontologies (KD-RDM and KD-Orange), but only the time for generating a workflow is measured. Hence, the generated workflows are not evaluated in terms of accuracy.

Most of WCEs allow the users to manually design workflows. They provide several tabs with various functions (e.g.,selecting and configuring operators). The data objects are implicitly represented as inputs or outputs of operators and can be set in the operator configuration views. Additionally, there are different operators for reading/writing different data formats. The data flows into/out of the operators through ports, which are visible. Then the users can manually drag an output from an operator to become an input for the next operator. However, they do not use abstract nodes for grouping the operators (except for some degree RapidMiner and IBM SPSS Modeler). Furthermore, the workflows are plain, rather than a hierarchy of tasks. IBM SPSS Modeler

has the concept of super nodes and sub-nodes. RapidMiner, on the other hand, uses a 'Subprocess', which contains an operator chain.

5.2.3. Workflow checking/repair. Both checking and repairing of workflows are important aspects of IDAs. Checking allows to discover errors at an early stage (before executing the workflow) and repairing fixes the problems discovered after the workflow execution. Therefore both save time when analyzing data.

Unfortunately, only a few of the existing IDAs can check the workflows for correctness and suggest possible solutions to fix the erroneous ones. This is one of the features encountered in some of the WCEs or in PDAS. For example, both RapidMiner and IBM SPSS Modeler offer meta-data propagation – the characteristics of the input data are available at any step, which helps to detect problems. RapidMiner 5.0 also supports the notion of "quick-fix", i.e. it suggests possible fixes for errors in the workflow design. In case the input data has missing values and the operator is not able to handle missing values, it will complain and suggest to use a 'replace missing values' operator.

PDAS use IOPEs for operators to ensure that they are used/applied in the right situations. However, they are not able to repair workflows. For example, if one workflow crashes during execution due to an error, no corrections are suggested.

Most of the existing IDAs are missing both features mainly as they were never tested by real users or as they do not offer support for an overall workflow (like ES, MLS), but also checking and repairing require more complexity.

5.2.4. Task decomposition vs. plain plan. Structuring the domain data into tasks and subtasks simplifies and improves the automatic generation of KDD workflows and therefore constitutes a desirable feature of IDAs. However only a few IDAs use this model. *Task-oriented user guidance* is implemented in CITRUS. Its user guidance module offers assistance in the selection and application of available techniques as well as the interpretation and evaluation of results. AIDE also uses hierarchical problem decomposition techniques: goals can be decomposed into several subgoals. Moreover, problem decomposition and abstraction constitute helpful features for the exploration in AIDE.

CRISP-DM [Chapman et al. 1999] follows a hierarchical process model having a set of tasks at four levels of abstraction: phase, generic task, specialized task, and process instance. The KDD-process consists of 6 phases, each of them comprising several generic tasks that cover all the possible data mining situations. The specialized tasks describe how the actions from the generic tasks should be accomplished in certain situations. The last level, namely process instance, represents a record of actions and results of a specific data mining operation. Tasks represent abstractions over several operators, for example the prediction to fill missing values and the prediction to predict the target have the same task. A similar approach is the one from eIDA which uses tasks and methods to guide the generation of workflows. This in turn reduces significantly the plan space and speeds up the plan generation.

5.2.5. Design support vs. explanations for result/output. IDAs, in general, provide either support for design or support for explanations. For an ideal IDA, both should be considered since they complement each other, designing an workflow is easier when useful explanations are present.

WCE enable users to manually design their workflows. Canvas-based tools include different types of design support. Besides the meta-data propagation and the quick-fixes, RapidMiner also employs descriptions of operators stored in a wiki. This wiki contains information about the algorithms, how they work, and descriptions of their important parameters. SAS Enterprise Miner has integrated debugging and runtime statistics. On the other hand, scripting-based tools provide help facilities with explanations about the functions. R and Matlab by themselves do not include design support,

however, their respective GUIs partially support it. HDMA offers some explanations based on the stored expert rules. IBM SPSS Statistics offers more support for *explanations* than other IDAs: the help menu provides extensive information about methods and algorithms even with examples illustrating the explained feature. Additionally, it features coaches that take you step-by-step through the process of interpreting the results or deciding which statistical analyses to choose via providing helpful examples. Hence, *learning by example* is a valuable feature of an IDA.

Opposed to WCEs, SES offer more support for explanations and interpretation of results. REX [Gale 1986] helps the user in *interpreting intermediate and final results* and gives useful instructions about statistical concepts. Springex has a primitive "why" explanation facility, which consists of a list of rules that have succeeded together with the conditions that have been asserted. However, the knowledge is unclear since it does not provide an explanation of technical terms, is superficial and incomplete. On the contrary Statistical Navigator uses an expert system with help and explanation facilities. Additionally, it has extensive reporting capabilities, including a short description of each technique and references to the literature and statistical packages that implement the technique. KENS and LMG provide explanations for concepts, but they do not handle the interpretation of results or explanations of the reasoning behind.

5.2.6. Experimental vs. analytical approach. Here we compare IDAs that can execute KDD-processes with the ones that only provide advice. Ideally an IDA should combine both execution and advice. The advantage of executing workflows is that the real performance comes from executing the process and all design support loses if the results of support cannot be directly used. However, only enumerating and executing many processes it is not feasible when a large number of processes/operators are present.

WCEs allow to execute workflows, as opposed to most ES which use underlying statistical packages for execution (e.g., REX is based on the S statistical system [Gale 1986]). Most of the PDAS rely on other data analysis tools or on web services for executing the generated workflows (eIDA, RDM, etc.). Relying on other WCEs or statistical packages can cause problems of scalability: the underlying package may change, evolve and therefore the assistance may need to be adjusted accordingly.

5.2.7. DM experience level. An ideal IDA should cover support for all types of users, however this is not a trivial feature and requires a more complex user support model. We defined four different categories of users: novice users – with only a little knowledge about ML algorithms, intermediary users – with a certain level of knowledge, expert users – with an extensive knowledge about the DM tasks, all – tools which address all types of users and unspecified – there is no evidence about the type of users. Most of the IDAs are meant to be used by expert or intermediary users. Only a few take novices into account and REX is one of these systems. However, KENS focuses on users with a certain level of knowledge. Inexperienced users can easily feel lost in SPRINGEX and Statistical Navigator since both systems offer a large amount of knowledge. MLT Consultant and CITRUS can be used by any domain expert. Current WCE systems have features that support naive users like explanations and help facilities. However, building workflows requires more effort and knowledge because of the large amount of operators. Some WCEs try to improve this process by providing on-the-fly error recognition, quick fixes, workflow templates, re-usable building blocks (e.g., RapidMiner, IBM SPSS Modeler).

5.2.8. Status and availability. IDAs are useful tools for data analysis, therefore they should be publicly available and maintained over the course of time. For the status we have Up-to-date (U) or Outdated (O) systems. WCEs are the most used since they offer a broad range of algorithms and are employed for analyzing real data. Most of

the other IDAs exist only as a proof of concept and were not designed to be kept up to date or applied to real-world tasks. However, these outdated systems have a great historical value because they were based on good ideas that are either reused in later systems (ES, MLS, CBR) or worth revisiting. For the availability metric, we use either (publicly) Available (A) or Not Available (NA). WCEs are all available and frequently used. Some older systems, such as DMA, MiningMart are also still available, even if they are no longer maintained, while some of the (younger) CBR and PDAS systems are not (yet) released for the public (e.g., HDMA and RDM). Even if these systems are currently not publicly available they are definitely worth exploring since they lead to better systems or emphasized an important characteristic of IDAs.

5.2.9. Final remarks. To conclude our comparison, we identify and explain the support limitations due to gaps in the systems' background knowledge. Considering Table I and Table II, we observe that many missing features in the support can be justified by the missing features in the background knowledge.

SES systems are completely lacking the concept of workflow, which can be easily explained by the fact that they only consider the DM and evaluation steps. They are missing the first steps of the KDD-process. This further explains missing features, like graphical editing of workflows, automatic execution, etc. Having a set of hardcoded rules at hand and focusing only on one single task, makes it easy to provide result interpretations and explanations for users.

Similarly, MLS systems, focus primarily on the DM step, i.e., mostly on classification and regression problems. The single step support is correlated with the presence of a predictive model, or in the case of ES, with the presence of rules. MLS employ the models to recommend the best suited method for a certain data set and task; on the other hand, ES use rules. If the system, however, does not produce predictive models, the support for single step is missing altogether. Featuring multiple steps does not necessarily mean that the system also provides support for multi-step KDD. For example, WCE allow to build workflows, but they do not provide sufficient guidance on the order of the operators. Table II suggests that if the IOPE are described, then the system is able to automatically generate workflows. Knowing when the operator can be applied and what it produces is essential for automated generation of workflows. This explains the workflow-related features of the PDAS. It is further a justification for the same missing features from the statistical and DM tools. Most of the WCEs allow multiple steps, but they have no description of operators' conditions and effects, therefore, they cannot decide when to apply an operator. However, current WCEs are improving and try to integrate such information.

Looking at the systems' status and availability we can argue that even if systems may analytically be ideal to solve a certain problem, they still need to be integrated into an useful, up-to-date environment to be of any use for the end user. Therefore, they should be integrated into an execution framework (e.g., as a WCE extension).

6. FUTURE DIRECTIONS

Looking back over the many IDAs discussed before, it becomes clear that each system uses its own limited amount of background knowledge to provide its own limited amount of assistance. Indeed, while the systems within each of the identified categories have continuously improved over earlier systems, there is very little cross-pollination between categories. For instance, looking at Table I, it is clear that no system uses both IOPE (for multi-step support) and predictive models (for single-step support). As a result, none offer a high level of support in both categories (see Table II). Also, while MLS and CBR systems use very detailed meta-data on the input data, PDAS and WCEs only use the bare minimum needed for workflow validation or

planning. This suggests that workflow generation can still be improved significantly by using more detailed meta-data to enrich the planning domain, or to find and adapt useful prior workflows.

In this section, we propose a framework for future IDAs that integrate the various forms of background knowledge to offer better and more complete support to end users.

6.1. IDA specification

This future IDA should combine the best aspects of all discussed systems:

Extensibility. Every IDA that covers a fixed number of techniques will at some point become obsolete. It is therefore important that new KD techniques can be added easily, without expert intervention, as in most MLS and WCE systems.

Self-maintenance. The IDA should be able to enrich its own background knowledge based on experience with new problems. For instance, the evaluations of all newly proposed workflows should be collected to further expand its background knowledge. In this way, it will automatically adapt to new types of data or new operators, as in most MLS and CBR systems.

Workflow execution. Ideally, the IDA should be able to execute proposed workflows, instead of just offering an abstract description. It therefore seems logical that the IDA is interoperable with, or integrated in WCE systems.

Ontologies. As in most modern IDAs in our survey, it seems useful to store prior knowledge about data analysis techniques in ontologies, since they allow information to be centralized, extended by many people, and automatically queried. Ideally, such an ontology would serve as a common vocabulary to describe workflows unambiguously.

Querying. We cannot foresee all questions a user may have. Therefore, it is best to structure the IDAs' background knowledge (e.g., in a database or ontology) so that it can be queried.

Planning. Workflow construction naturally translates to a planning problem, as many operators only work on certain types of data. Especially when a workflow needs to be generated from scratch, planning is an indispensable component of an ideal IDA.

Learning. Last but not least, the IDA should get better as it gains more experience. This can be done by collecting meta-data and modeling it, as in MLS, or use it to compute heuristics (e.g., operator speed) for use in heuristic planners that will improve over time.

6.2. IDA Architecture

These aspects are combined in the IDA architecture shown in Figure 7:

Input. The user first feeds the input data into the IDA and defines requirements for the DM workflow. The IDA then interprets the data and extract all properties that are needed for subsequent support, e.g., measuring statistical properties.

Background knowledge. The IDA stores its background knowledge in a public, online collaborative knowledge base (the cloud on the left of Figure 7) to which many users can contribute information. It is structured in an ontology or database so that the IDA can query it for specific information. Users should also be able to query it, and upload new workflows directly from WCE systems. We will discuss this in more detail below.

Planning. The IDA queries this knowledge base for detailed information about available operators, or any prior workflows that seem useful for the given problem. With this information, it will define planning problems, either to build workflows from scratch or 'fix' prior workflows (as in NeXT). Then, it will use an AI planner to build the most useful plans, and rank them according to the user's preferences.

Adaptation and execution. The user can then inspect, alter and execute the proposed workflows. WCEs already offer extensive support for this (e.g. , graphical workflow editing), so the IDA can interface with such systems to delegate these tasks. Using an

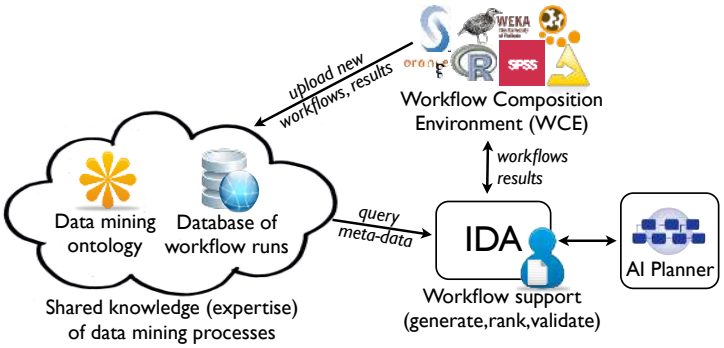

Fig. 7: A proposed platform for intelligent KD support.

interface allows the IDA to support multiple WCEs, although it can also be seamlessly integrated into one particular WCE. Workflow execution can also be initiated by the IDA itself, for instance as part of the ranking procedure when the user imposes hard constraints on the speed or outcome of the workflow. At any time, the user should be able to remove or add requirements to (parts of) the workflow and reinitiate workflow generation.

6.2.1. A collaborative platform. As more and more KDD techniques are introduced, it becomes infeasible for experts to keep everything up-to-date. It seems therefore crucial to take a *collaborative* approach, in which all users can upload and share their workflows, evaluation results, ontological descriptions, and other forms of background knowledge with IDAs and the community at large.

In its simplest form, this knowledge base is an open ontology which users can update with descriptions of new operators. Ideally, however, it also acts as a *workflow repository*, containing both abstract descriptions of workflows (i.e., workflow *templates*), as concrete workflow *runs* on specific datasets, including meta-data of those datasets, and meta-data on the workflow's execution (e.g., runtimes and other evaluation metrics). This will create an integrated source of background knowledge to drive the various types of support put forward in this survey.

This repository can be implemented as a database, e.g. an RDF triple store or a relational database. RDF triple stores seem the most obvious choice since they can directly store ontological descriptions of operators and even entire workflows, exchange information in the RDF format, and allow advanced querying with SPARQL queries. Relational databases, on the other hand, are more familiar to many people, and most WCEs offer direct support for them. In this case, a common DM ontology should still be used to define a controlled vocabulary, so that workflow descriptions can be expressed in an unambiguous way.

Such collaborative knowledge bases are common practice in many other sciences. In bioinformatics, for instance, *microarray databases* [Stoeckert et al. 2002] collect gene expression data described according to the MicroArray Gene Expression ontology (MAGE-MO) [Stoeckert et al. 2002] and the Gene Ontology [Ashburner et al. 2000]. Astronomical observations from telescopes all over the world are collected in so-called Virtual Observatories [Szalay and Gray 2001], also using informal ontologies [Derriere et al. 2006], and in physics, low-energy nuclear reaction data is collected in ENDF

libraries[18], and high-energy particle physics data on the HEPDATA[19] website. Given that DM is performed entirely *in-silico*, it should even be easier to develop similar repositories for DM workflows.

A requirement for such a collaborative system is that workflows can be shared in a standardized format that includes all details needed to reuse, or even rerun the workflows. This ensures interoperability with many different WCEs, which currently each use their own formats to describe workflows and operators. Using a standardized format, WCEs could easily export/share all constructed workflows and their evaluations, and import/reuse workflows that were created by different users. As in the other sciences mentioned above, an ontology of DM concepts and techniques will be very useful to share and organize workflows uniformly, and even map imported workflows to local implementations or web services. The latter are also ensure greater extensibility [Podpečan et al. 2011].

6.3. Implementation

While such an IDA may still be some way off, recently, a great deal of work has been done that brings us a lot closer to realizing it.

6.3.1. Taverna and myExperiment. Taverna [Roure et al. 2009] is a platform designed to help scientists compose executable workflows in which the components are web services, especially for biological experiments. Similarly, Triana [Taylor et al. 2007] supports workflow execution in peer-to-peer networks and the Grid. myExperiment [Goble et al. 2010] is a collaborative environment where scientists can publish their workflows and experiment plans, share them with groups and find those of others. It is a more open platform, where workflows can be uploaded in a variety of formats, mostly XML-based. Also, flat files can be attached that contain, for instance, the workflow's output.

While mostly aimed at bio-informatics, these platforms could be used to share more general data mining workflows as well. On the other hand, they do not organize all data in a way that supports the detailed queries needed by an IDA.

6.3.2. Experiment Databases. Experiment databases (ExpDBs) [Blockeel and Vanschoren 2007; Vanschoren et al. 2012] are databases specifically designed to collect huge numbers of detailed DM algorithm evaluations, and to automatically organize all results so they can be thoroughly queried. They allow users to upload (share) workflows together with the actual results of those workflows, so that they can be reused by other users. An ExpDB for supervised classification is available online at `http://expdb.cs.kuleuven.be`. An ontology for DM experimentation, called Exposé, is used to define a formal language for sharing these workflows, as well as to structure the database. Moreover, based on this ontology, an XML-based, extensible workflow description format is provided (ExpML) that can be used to export workflows from any WCE and store them in the database.

The system also offers online query interfaces that allow researchers to quickly filter through over 650,000 experiments, ask very specific questions about learning performance, and visualize the returned results. By carefully storing and organizing experiments, their results can be browsed and analyzed from different viewpoints and reused in many ways. For instance, they can be used to estimate an operator's sensitivity to noise, or build predictive models that propose DM techniques based on dataset properties. It also calculates and stores a wide range of data properties for all submitted datasets.

[18]http://www.nndc.bnl.gov/exfor/endf00.jsp
[19]http://durpdg.dur.ac.uk/hepdata/

6.3.3. Ontologies. While there is no universally established DM ontology yet, there are several DM ontologies currently under development, such as Exposé [Vanschoren et al. 2012], OntoDM [Panov et al. 2009], DMOP [Hilario et al. 2009], DMWF [Kietz et al. 2009], KDDONTO [Diamantini et al. 2009a] and KD ontology [Žáková et al. 2010]. Some of these are complementary, some overlap. To coordinate efforts, an open development forum[20] (similar to the OBO foundry[21]) has been created, where everyone can browse the latest DM ontologies and propose changes and extensions. Alternatively, *semantic wikis* [Boulos 2009] could be set up to collaboratively edit DM ontologies.

6.3.4. Planning. Concerning planning, several systems already translate the ontological descriptions of KDD operators to planning domain descriptions. Some query the ontologies before starting the actual planning process [Klusch et al. 2005; Liu et al. 2007; Sirin et al. 2004], others integrate a reasoning engine in the planner, so that it can directly query the ontology when needed [Kietz et al. 2009; Žáková et al. 2010].

Klusch et al. [Klusch et al. 2005] and Liu et al. [Liu et al. 2007] use a classical STRIPS planner to produce the planning, Sirin et al. [Sirin et al. 2004] and Kietz et al. [Kietz et al. 2009] propose an Hierarchical Task Network (HTN) planning approach [Sacerdoti 1974], and Žáková et al. [Žáková et al. 2010] use an adaptation of the heuristic Fast-Forward (FF) planning system [Hoffmann and Nebel 2001].

Finally, Kalousis et al. [Kalousis et al. 2008] propose a system that combines planning and meta-learning. It contains a probabilistic meta-learner which dynamically adjusts transition probabilities between DM operators, conditioned on the current application task and data, user-specified performance criteria, quality scores of workflows applied in the past to similar tasks and data, and the users profile. Thus, as more workflows are stored as meta-knowledge, and more is known about the users building those workflows, it will learn to build workflows that better fit to the user.

7. CONCLUSIONS

KDD has transformed into a mature field and the data analysis approaches provided by it have ubiquitously been included into many everyday applications such as SPAM filtering or credit card fraud analysis. The process of actually performing data analysis is oftentimes more of an art than a science – beginners are startled by the plethora of operators and specialists limit their activity to few known approaches. Hence, data analysts need to be thoroughly supported in their work.

The present survey analyzes the intelligent support provided by data analysis—systems which are called intelligent discovery assistants (IDAs)—tools from their beginning (usually as some form of expert systems) until today (as WCE). These are systems which incorporate different types of support and advice to facilitate the data analysis process. We structure the analysis of IDA approaches from the last three decades based on the kinds of background knowledge they use and the types of support they provide. Based on criteria along these two perspectives on analysis, we provide a thorough comparison of the selected systems identifying their advantages and drawbacks. This leads the way to the identification of possible future directions and provides first examples of these novel approaches.

In summary, as the exploration of data becomes increasingly important in today's scientific and industrial settings the need for automated support for data analysis is likely to increase. This summarized overview of the systems provided in this survey helps readers to quickly learn about the strengths and weaknesses in this field. As such, it provides a major building block for creating future studies.

[20]The Data Mining Ontology Foundry: http://www.dmo-foundry.org
[21]The Open Biological and Biomedical Ontologies: http://www.obofoundry.org

REFERENCES

AHA, D. W. 1992. Generalizing from case studies: a case study. In *Proceedings of the ninth international workshop on Machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1–10.

AMANT, R. AND COHEN, P. 1998a. Interaction with a mixed-initiative system for exploratory data analysis. *Knowledge-Based Systems 10,* 5, 265–273.

AMANT, R. S. AND COHEN, P. 1998b. Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics 7,* 4, 545–558.

ASHBURNER, M., BALL, C., BLAKE, J., BOTSTEIN, D., BUTLER, H., CHERRY, J., DAVIS, A., DOLINSKI, K., DWIGHT, S., EPPIG, J., HARRIS, M., HILL, D., ISSEL-TARVER, L., KASARSKIS, A., LEWIS, S., MATESE, J., RICHARDSON, J., RINGWALD, M., RUBIN, G., AND SHERLOCK, G. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics 25*, 25–29.

BENSUSAN, H. AND KALOUSIS, A. 2001. Estimating the predictive accuracy of a classifier. *Lecture Notes in Computer Science 2167*, 25–36.

BERNSTEIN, A. AND DAENZER, M. 2007. The NExT system: Towards true dynamic adaptations of semantic web service compositions. *Lecture Notes in Computer Science 4519*, 739–748.

BERNSTEIN, A., PROVOST, F., AND HILL, S. 2005. Toward intelligent assistance for a data mining process: an ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering 17,* 4, 503–518.

BERTHOLD, M. R., CEBRON, N., DILL, F., GABRIEL, T. R., KÖTTER, T., MEINL, T., OHL, P., THIEL, K., AND WISWEDEL, B. 2009. Knime - the konstanz information miner: version 2.0 and beyond. *SIGKDD Explor. Newsl. 11*, 26–31.

BLOCKEEL, H. AND VANSCHOREN, J. 2007. Experiment databases: Towards an improved experimental methodology in machine learning. *Lecture Notes in Computer Science 4702*, 6–17.

BLUM, A. AND FURST, M. 1997. Fast planning through planning graph analysis* 1. *Artificial intelligence 90,* 1-2, 281–300.

BOTIA, J., GOMEZ-SKARMETA, A., VALDES, M., AND PADILLA, A. 2001. METALA: A meta-learning architecture. *Lecture Notes in Computer Science 2206*, 688–698.

BOULOS, M. N. K. 2009. Semantic Wikis: A Comprehensible Introduction with Examples from the Health Sciences. *Journal of Emerging Technologies in Web Intelligence*.

CASTIELLO, C., CASTELLANO, G., AND FANELLI, A. 2005. Meta-data: Characterization of input features for meta-learning. *Modeling Decisions for Artificial Intelligence 3558*, 457–468.

CASTIELLO, C. AND FANELLI, A. 2005. Meta-learning experiences with the mindful system. *Lecture Notes in Artificial Intelligence 3801*, 321–328.

CERRITO, P. 2007. *Introduction to Data Mining Using SAS Enterprise Miner*. SAS Publishing, NC, USA.

CHANDRASEKARAN, B., JOHNSON, T., AND SMITH, J. 1992. Task-structure analysis for knowledge modeling. *Communications of the ACM 35,* 9, 124–137.

CHANDRASEKARAN, B. AND JOSEPHSON, J. 1999. What are ontologies, and why do we need them? *IEEE Intelligent Systems 14,* 1, 20–26.

CHAPMAN, P., CLINTON, J., KHABAZA, T., REINARTZ, T., AND WIRTH, R. 1999. The crisp-dm process model. *The CRIP–DM Consortium 310*.

CHAREST, M., DELISLE, S., CERVANTES, O., AND SHEN, Y. 2008. Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *Intelligent Data Analysis 12*, 1–26.

CHOINSKI, M. AND CHUDZIAK, J. 2009. Ontological learning assistant for knowledge discovery and data mining. In *Computer Science and Information Technology*. IEEE, 147–155.

CRAW, S., SLEEMAN, D., GRANER, N., AND RISSAKIS, M. 1992. Consultant: Providing advice for the machine learning toolbox. In *Proceedings of the Annual Technical Conference on Expert Systems (ES)*. 5–23.

DERRIERE, S., PREITE-MARTINEZ, A., AND RICHARD, A. 2006. UCDs and ontologies. *ASP Conference Series 351*, 449.

DIAMANTINI, C., POTENA, D., AND STORTI, E. 2009a. KDDONTO: An ontology for discovery and composition of KDD algorithms. In *Proceedings of the ECML-PKDD'09 Workshop on Service-oriented Knowledge Discovery*. 13–24.

DIAMANTINI, C., POTENA, D., AND STORTI, E. 2009b. Ontology-driven KDD process composition. *Lecture Notes in Computer Science 5772*, 285–296.

ENGELS, R. 1996. Planning tasks for knowledge discovery in databases: Performing task-oriented user-guidance. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD)*, 170–175.

ENGELS, R., LINDNER, G., AND STUDER, R. 1997. A guided tour through the data mining jungle. In *Proceedings of the 3nd International Conference on Knowledge Discovery in Databases*. AAAI Press, Menlo Park, CA, USA, 163–166.

EROL, K. 1996. Hierarchical task network planning: formalization, analysis, and implementation. Ph.D. thesis, University of Maryland at College Park, College Park, MD, USA. UMI Order No. GAX96-22054.

FAYYAD, U., PIATETSKY-SHAPIRO, G., AND SMYTH, P. 1996. From data mining to knowledge discovery in databases. *AI magazine 17,* 3, 37–54.

FOX, M. AND LONG, D. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research 20,* 1, 61–124.

GALE, W. 1986. Rex review. In *Artificial intelligence and statistics*. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, 173–227.

GIRAUD-CARRIER, C. 2005. The data mining advisor: meta-learning at the service of practitioners. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*. IEEE Computer Society, 113–119.

GOBLE, C., BHAGAT, J., ALEKSEJEVS, S., CRUICKSHANK, D., MICHAELIDES, D., NEWMAN, D., BORKUM, M., BECHHOFER, S., ROOS, M., LI, P., AND DE ROURE, D. 2010. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucl. Acids Res.*.

GOEBEL, M. AND GRUENWALD, L. 1999. A survey of data mining and knowledge discovery software tools. *SIGKDD Explor. Newsl. 1*, 20–33.

GRABCZEWSKI, K. AND JANKOWSKI, N. 2007. Versatile and efficient meta-learning architecture: Knowledge representation and .... *IEEE Symposium on Computational Intelligence and Data Mining*, 51–58.

GRANER, N., SHARMA, S., SLEEMAN, D., RISSAKIS, M., CRAW, S., AND MOORE, C. 1993. The machine learning toolbox consultant. *International Journal on AI Tools 2,* 3, 307–328.

GRIMMER, U. 1996. Clementine: Data mining software. *Classification and Multivariate Graphics: Models, Software and Applications*, 25–31.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. 2009. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter 11,* 1, 10–18.

HAND, D. 1985. Statistical expert systems: necessary attributes. *Journal of applied Statistics 12,* 1, 19–27.

HAND, D. 1987. A statistical knowledge enhancement system. *Journal of the Royal Statistical Society. Series A (General) 150,* 4, 334–345.

HAND, D. 1990. Practical experience in developing statistical knowledge enhancement systems. *Annals of Mathematics and Artificial Intelligence 2,* 1, 197–208.

HAND, D. 1997. Intelligent data analysis: issues and opportunities. In *Advances in Intelligent Data Analysis. Reasoning about Data: Second International Symposium, IDA-97, London, UK, August 1997. Proceedings*. IDA '97. Springer-Verlag, London, UK, 1–14.

HERNANSAEZ, J., BOTA, J., AND SKARMETA, A. 2004. METALA: a J2EE technology based framework for web mining. *Revista Colombiana de Computación 5,* 1.

HILARIO, M. AND KALOUSIS, A. 2001. Fusion of meta-knowledge and meta-data for case-based model selection. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. PKDD '01. Springer-Verlag, London, UK, 180–191.

HILARIO, M., KALOUSIS, A., NGUYEN, P., AND WOZNICA, A. 2009. A data mining ontology for algorithm selection and meta-mining. In *Proceedings of the ECML-PKDD'09 Workshop on Service-oriented Knowledge Discovery*. 76–87.

HOFFMANN, J. AND NEBEL, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research 14*, 253–302.

HORROCKS, I., PATEL-SCHNEIDER, P., AND BOLEY, H. 2004. SWRL: A semantic web rule language combining OWL and RuleML.

IHAKA, R. AND GENTLEMAN, R. 1996. R: A language for data analysis and graphics. *Journal of computational and graphical statistics 5,* 3, 299–314.

KALOUSIS, A. 2002. Algorithm selection via meta-learning. Ph.D. thesis.

KALOUSIS, A., BERNSTEIN, A., AND HILARIO, M. 2008. Meta-learning with kernels and similarity functions for planning of data mining workflows. In *Proceedings of the ICML/UAI/COLT'08 Workshop on Planning to Learn*. 23–28.

KALOUSIS, A. AND HILARIO, M. 2001. Model selection via meta-learning: a comparative study. *International Journal on Artificial Intelligence Tools 10,* 4, 525–554.

KALOUSIS, A. AND THEOHARIS, T. 1999. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis 3,* 4, 319–337.

KIETZ, J., SERBAN, F., AND BERNSTEIN, A. 2010. eProPlan: A Tool to Model Automatic Generation of Data Mining Workflows. In *3 rd PLANNING TO LEARN WORKSHOP WS9 AT ECAI 2010*. 15.

KIETZ, J., SERBAN, F., BERNSTEIN, A., AND FISCHER, S. 2009. Towards cooperative planning of data mining workflows. In *Proceedings of the ECML-PKDD'09 Workshop on Service-oriented Knowledge Discovery*. 1–12.

KIETZ, J., VADUVA, A., AND ZÜCKER, R. 2000. Mining mart: combining case-based-reasoning and multistrategy learning into a framework to reuse kdd-application. In *Proceedings of the fifth International Workshop on Multistrategy Learning (MSL2000). Guimares, Portugal*. Vol. 311.

KLUSCH, M., GERBER, A., AND SCHMIDT, M. 2005. Semantic web service composition planning with OWLS-Xplan. In *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web*. AAAI Press, Menlo Park, California, USA, 55–62.

KODRATOFF, Y., SLEEMAN, D., USZYNSKI, M., CAUSSE, K., AND CRAW, S. 1992. Building a machine learning toolbox. *In: Enhancing the knowledge engineering process: contributions from ESPRIT (eds. L Steels and B Lepape). Elsevier*, 81–108.

KOHAVI, R., BRODLEY, C. E., FRASCA, B., MASON, L., AND ZHENG, Z. 2000. Kdd-cup 2000 organizers' report: peeling the onion. *SIGKDD Explor. Newsl. 2*, 86–93.

LEITE, R. AND BRAZDIL, P. 2007. An iterative process for building learning curves and predicting relative performance of classifiers. *Lecture Notes in Computer Science 4874*, 87–98.

LEVESQUE, R. 2005. *SPSS programming and data management: A guide for SPSS and SAS users*. SPSS, Chicago, USA.

LINDNER, G. AND STUDER, R. 1999. AST: Support for algorithm selection with a CBR approach. *Lecture Notes in Computer Science 1704*, 418–423.

LIU, Z., RANGANATHAN, A., AND RIABOV, A. 2007. A planning approach for message-oriented semantic web service composition. *Proceedings of the AAAI National Conference On Artificial Intelligence 5,* 2, 1389–1394.

MATHWORKS, I. 2004. Matlab. *The MathWorks, Natick, MA*.

MCDERMOTT, D., GHALLAB, M., HOWE, A., KNOBLOCK, C., RAM, A., VELOSO, M., WELD, D., AND WILKINS, D. 1998. PDDL-the planning domain definition language.

MICHIE, D., SPIEGELHALTER, D., AND TAYLOR, C. 1994. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA.

MIERSWA, I., WURST, M., KLINKENBERG, R., SCHOLZ, M., AND EULER, T. 2006. Yale: Rapid prototyping for complex data mining tasks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 935–940.

MIKUT, R. AND REISCHL, M. 2011. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, n/a–n/a.

MORIK, K. AND SCHOLZ, M. 2004. The MiningMart approach to knowledge discovery in databases. *In: Intelligent Technologies for Information Analysis (eds. N. Zhong, J. Liu), Springer*, 47–65.

NONAKA, I. AND TAKEUCHI, H. 1995. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, New York, USA.

OINN, T., ADDIS, M., FERRIS, J., MARVIN, D., GREENWOOD, M., CARVER, T., POCOCK, M., WIPAT, A., AND LI, P. 2004. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics 20,* 17, 3045–3054.

PANOV, P., SOLDATOVA, L., AND DŽEROSKI, S. 2009. Towards an ontology of data mining investigations. *Lecture Notes in Artificial Intelligence 5808*, 257–271.

PATEL-SCHNEIDER, P., HAYES, P., HORROCKS, I., ET AL. 2004. OWL web ontology language semantics and abstract syntax. *W3C recommendation 10*.

PENG, Y., FLACH, P., BRAZDIL, P., AND SOARES, C. 2002. Decision tree-based data characterization for meta-learning. In *Proceedings of the ECML-PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*. 111–122.

PENG, Y., FLACH, P., SOARES, C., AND BRAZDIL, P. 2002. Improved dataset characterisation for meta-learning. *Lecture Notes in Computer Science 2534*, 141–152.

PFAHRINGER, B., BENSUSAN, H., AND GIRAUD-CARRIER, C. 2000. Meta-learning by landmarking various learning algorithms. *Proceedings of the International Conference on Machine Learning (ICML) 951,* 2000, 743–750.

PODPEČAN, V., ZEMENOVA, M., AND LAVRAČ, N. 2011. Orange4ws environment for service-oriented data mining. *The Computer Journal*.

RAES, J. 1992. Inside two commercially available statistical expert systems. *Statistics and Computing 2,* 2, 55–62.

RENDELL, L., SESHU, R., AND TCHENG, D. 1987. Layered concept learning and dynamically-variable bias management. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 308–314.

RICE, J. 1976. The algorithm selection problem. *Advances in Computers 15*, 65–118.

ROURE, D. D., GOBLE, C., AND STEVENS, R. 2009. The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems 25*, 561–567.

RUSSELL, D. M., STEFIK, M. J., PIROLLI, P., AND CARD, S. K. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*. CHI '93. ACM, New York, NY, USA, 269–276.

SACERDOTI, E. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence 5,* 2, 115–135.

SCHAFFER, C. 1994. A conservation law for generalization performance. In *International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 259–265.

SIRIN, E. AND PARSIA, B. 2007. SPARQL-DL: SPARQL query for OWL-DL. In *Proceedings of the International Workshop on OWL Experiences and Directions (OWLED)*.

SIRIN, E., PARSIA, B., GRAU, B., KALYANPUR, A., AND KATZ, Y. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web 5,* 2, 51–53.

SIRIN, E., PARSIA, B., WU, D., HENDLER, J., AND NAU, D. 2004. HTN planning for web service composition using SHOP2. *Journal of Web Semantics 1,* 4, 377–396.

SLEEMAN, D., RISSAKIS, M., CRAW, S., GRANER, N., AND SHARMA, S. 1995. Consultant-2: Pre-and post-processing of machine learning applications. *International Journal of Human Computer Studies 43,* 1, 43–63.

SMITH-MILES, K. 2008. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys 41,* 1, Article 6.

SOARES, C., BRAZDIL, P., AND KUBA, P. 2004. A meta-learning method to select the kernel width in support vector regression. *Machine Learning 54*, 195–209.

STOECKERT, C., CAUSTON, H., AND BALL, C. 2002. Microarray databases: standards and ontologies. *Nature Genetics 32*, 469–473.

SZALAY, A. AND GRAY, J. 2001. The world-wide telescope. *Science 293*, 2037–2040.

TAYLOR, I., SHIELDS, M., WANG, I., AND HARRISON, A. 2007. The Triana workflow environment: Architecture and applications. In *In: Workflows for e-Science (eds. I. Taylor, E. Deelman, D. Gannon, m. Shields), Springer*. Springer, London, UK, 320–339.

TODOROVSKI, L., BLOCKEEL, H., AND DŽEROSKI, S. 2002. Ranking with predictive clustering trees. *Lecture Notes in Computer Science 2430*, 444–455.

VANSCHOREN, J. 2010. Understanding machine learning performance with experiment databases. Ph.D. thesis, Katholieke Universiteit Leuven.

VANSCHOREN, J., BLOCKEEL, H., PFAHRINGER, B., AND HOLMES, G. 2012. Experiment databases: A new way to share, organize and learn from experiments. *Machine Learning* In press, DOI 10.1007/s10994-011-5277-0.

VILALTA, R. AND DRISSI, Y. 2002a. A characterization of difficult problems in classification. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*. CSREA Press, Las Vegas, USA, 133–138.

VILALTA, R. AND DRISSI, Y. 2002b. A perspective view and survey of meta-learning. *Artif. Intell. Rev. 18*, 77–95.

WIRTH, R., SHEARER, C., GRIMMER, U., REINARTZ, T., SCHLOSSER, J., BREITNER, C., ENGELS, R., AND LINDNER, G. 1997. Towards process-oriented tool support for knowledge discovery in databases. *Lecture Notes in Computer Science 1263*, 243–253.

WOLPERT, D. 2001. The supervised learning no-free-lunch theorems. In *Proceedings of the Online World Conference on Soft Computing in Industrial Applications*. 25–42.

YANG, G., KIFER, M., ZHAO, C., AND CHOWDHARY, V. 2002. *Flora-2: Users manual*. Department of Computer Science, Stony Brook University, Stony Brook, USA.

ŽÁKOVÁ, M., KŘEMEN, P., ŽELEZNÝ, F., AND LAVRAČ, N. 2010. Automatic knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering online 1st*, 53–264.