Journal of Artificial Intelligence Research 51 (2014) Submitted 02/14; published 11/14

Using Meta-mining to Support Data Mining Workflow Planning and Optimization

Phong Nguyen PHONG.NGUYEN@UNIGE.CH
Melanie Hilario MELANIE.HILARIO@UNIGE.CH
Department of Computer Science
University of Geneva
Switzerland

Alexandros Kalousis ALEXANDROS.KALOUSIS@HESGE.CH
Department of Business Informatics
University of Applied Sciences
Western Switzerland, and
Department of Computer Science
University of Geneva
Switzerland

Abstract

Knowledge Discovery in Databases is a complex process that involves many different data processing and learning operators. Today's Knowledge Discovery Support Systems can contain several hundred operators. A major challenge is to assist the user in designing workflows which are not only valid but also – ideally – optimize some performance measure associated with the user goal. In this paper we present such a system. The system relies on a meta-mining module which analyses past data mining experiments and extracts meta-mining models which associate dataset characteristics with workflow descriptors in view of workflow performance optimization. The meta-mining model is used within a data mining workflow planner, to guide the planner during the workflow planning. We learn the meta-mining models using a similarity learning approach, and extract the workflow descriptors by mining the workflows for generalized relational patterns accounting also for domain knowledge provided by a data mining ontology. We evaluate the quality of the data mining workflows that the system produces on a collection of real world datasets coming from biology and show that it produces workflows that are significantly better than alternative methods that can only do workflow selection and not planning.

1. Introduction

Learning models and extracting knowledge from data using data mining can be an extremely complex process which requires combining a number of Data Mining (DM) operators, selected from large pools of available operators, combined into a data mining workflow. A DM workflow is an assembly of individual data transformations and analysis steps, implemented by DM operators, composing a DM process with which a data analyst chooses to address his/her DM task. Workflows have recently emerged as a new paradigm for representing and managing complex computations accelerating the pace of scientific progress. Their (meta-)analysis is becoming increasingly challenging with the growing number and complexity of available operators (Gil et al., 2007).

Today's second generation knowledge discovery support systems (KDSS) allow complex modeling of workflows and contain several hundreds of operators; the RapidMiner platform (Klinkenberg, Mierswa, & Fischer, 2007), in its extended version with Weka (Hall et al., 2009) and R (R Core Team, 2013), proposes actually more than 500 operators, some of which can have very complex data and control flows, e.g. bagging or boosting operators, in which several sub-workflows are interleaved. As a consequence, the possible number of workflows that can be modeled within these systems is on the order of several millions, ranging from simple to very elaborated workflows with several hundred operators. Therefore the data analyst has to carefully select among those operators the ones that can be meaningfully combined to address his/her knowledge discovery problem. However, even the most sophisticated data miner can be overwhelmed by the complexity of such modeling, having to rely on his/her experience and biases as well as on thorough experimentation in the hope of finding the best operator combination. With the advance of new generation KDSS that provide even more advanced functionalities, it becomes important to *provide automated support to the user in the workflow modeling process*, an issue that has been identified as one of the top-ten challenges in data mining (Yang & Wu, 2006).

2. State of the Art in DM Workflow Design Support

During the last decade, a rather limited number of systems have been proposed to provide automated user support in the design of DM workflows. Bernstein, Provost, and Hill (2005) propose an ontology-based *Intelligent Discovery Assistant* (IDA) that plans valid DM workflows – valid in the sense that they can be executed without any failure – according to basic descriptions of the input dataset such as attribute types, presence of missing values, number of classes, etc. By describing into a DM ontology the input conditions and output effects of DM operators, according to the three main steps of the KD process, pre-processing, modeling and post-processing (Fayyad, Piatetsky-Shapiro, & Smyth, 1996), IDA systematically enumerates with a workflow planner all possible valid operator combinations, workflows, that fulfill the data input request. A ranking of the workflows is then computed according to user defined criteria such as speed or memory consumption which are measured from past experiments.

Záková, Kremen, Zelezny, and Lavrac (2011) propose the KD ontology to support automatic design of DM workflows for relational DM. In this ontology, DM relational algorithms and datasets are modeled with the semantic web language OWL-DL, providing semantic reasoning and inference in querying over a DM workflow repository. Similar to IDA, the KD ontology describes DM algorithms with their data input/output specifications. The authors have developed a translator from their ontology representation to the *Planning Domain Definition Language* (PDDL) (McDermott et al., 1998), with which they can produce abstract directed-acyclic graph workflows using a FF-style planning algorithm (Hoffmann, 2001). They demonstrate their approach on genomic and product engineering (CAD) use-cases where complex workflows are produced that make use of relational data structure and background knowledge.

More recently, the e-LICO project[1] featured another IDA built upon a planner which constructs DM plans following a hierarchical task networks (HTN) planning approach. The

1. http://www.e-lico.eu

specification of the HTN is given in the *Data Mining Workflow* (DMWF) ontology (Kietz, Serban, Bernstein, & Fischer, 2009). As its predecessors, the e-LICO IDA has been designed to identify operators whose preconditions are met at a given planning step in order to plan valid DM workflows and does an exhaustive search in the space of possible DM plans.

None of the three DM support systems that we have just discussed consider the eventual performance of the workflows they plan with respect to the DM task that they are supposed to address. For example, if our goal is to plan/design workflows that solve a classification problem, we would like to consider a measure of classification performance, such as accuracy, and deliver workflows that optimize it. All the discussed DM support systems deliver an extremely large number of plans, DM workflows, which are typically ranked with simple heuristics, such as workflow complexity or expected execution time, leaving the user at a loss as to which is the best workflow in terms of the expected performance in the DM task that he/she needs to address. Even worse, the planning search space can be so large that the systems can even fail to complete the planning process, see for example the discussion by Kietz, Serban, Bernstein, and Fischer (2012).

There has been considerable work that tries to support the user, in view of performance maximization, for a very *specific* part of the DM process, that of modeling or learning. A number of approaches have been proposed, collectively identified as meta-learning (Brazdil, Giraud-Carrier, Soares, & Vilalta, 2008; Kalousis, 2002; Kalousis & Theoharis, 1999; Hilario, 2002; Soares & Brazdil, 2000). The main idea in meta-learning is that given a new dataset the system should be able to rank a pool of learning algorithms with respect to their expected performance on the dataset. To do so, one builds a meta-learning model from the analysis of past learning experiments, searching for associations between algorithms' performances and dataset characteristics. However, as already mentioned, all meta-learning approaches address only the learning/modeling part and are not applicable on the complete process level.

In the work of Hilario, Nguyen, Do, Woznica, and Kalousis (2011), we did a first effort to lift meta-learning ideas to the level of complete DM workflows. We proposed a novel meta-learning framework that we called *meta-mining* or process-oriented meta-learning applied on the complete DM process. We associate workflow descriptors and dataset descriptors, applying decision tree algorithms on past experiments, in order to learn which couplings of workflows and datasets will lead to high predictive performance. The workflow descriptors were extracted using frequent pattern mining accommodating also background knowledge, given by the *Data Mining Optimization* (DMOP) ontology, on DM tasks, operators, workflows, performance measures and their relationships. However the predictive performance of the system was rather low, due to the limited capacity of decision trees in capturing the relations between the dataset and the workflow characteristics that were essential for performance prediction.

To address the above limitation, we presented in the work of Nguyen, Wang, Hilario, and Kalousis (2012b) an approach that learns what we called heterogeneous similarity measures, associating dataset and workflow characteristics. There we learn similarity measures in the dataset space, the workflow space, and the dataset-workflow space. These similarity measures reflect respectively: the similarity of the datasets as it is given by the similarity of the relative workflow performance vectors of the workflows that were applied on them; the similarity of the workflows given by their performance based similarity on different datasets;

the dataset-workflow similarity based on the expected performance of the latter applied on the former. However the two meta-mining methods described here were limited to select from, or rank, a set of given workflows according to their expected performance, i.e. they cannot plan new workflows given an input dataset.

Retrospectively, we presented in the work of Nguyen, Kalousis, and Hilario (2011) an initial blueprint of an approach that does DM workflow planning in view of workflow performance optimization. There we suggested that the planner should be guided by a meta-mining model that ranks partial candidate workflows at each planning step. We also gave a preliminary evaluation of the proposed approach with interesting results (Nguyen, Kalousis, & Hilario, 2012a). However, the meta-mining module was rather trivial, it uses dataset and pattern-based workflow descriptors and does a nearest-neighbor search over the dataset descriptors to identify the most similar datasets to the dataset for which we want to plan the workflows. Within that neighborhood, it ranks the partial workflows using the support of the workflow patterns on the workflows that perform best on the datasets of the neighborhood. The pattern-based ranking of the workflows was cumbersome and heuristic; the system was not learning associations of dataset and workflow characteristics which explicitly optimize the expected workflow performance, which is what must guide the workflow planning. This approach was then deployed by Kietz et al. (2012).

In this paper, we follow the line of work we first sketched in the work of Nguyen et al. (2011). We couple tightly together a workflow planning and a meta-mining module to develop a DM workflow planning system that given an input dataset designs workflows that are expected to optimize the performance on the given dataset. The meta-mining module applies the heterogeneous similarity learning method presented by Nguyen et al. (2012b) to learn associations between dataset and workflow descriptors that lead to optimal performance. It exploits then the learned associations to guide the planner in the workflow construction during the planning. To the best of our knowledge, it is the first system of its kind, i.e. being able to design DM workflows that are specifically tailored to the characteristics of the input dataset in view of optimizing a DM task performance measure. We evaluate the system on a number of real world datasets and show that the workflows it plans are significantly better than the workflows delivered by a number of baseline methods.

The rest of this paper is structured as follows. In section 3, we present the global architecture of the system, together with a brief description of the planner. In section 4 we describe in detail the meta-mining module, including the dataset and workflow characteristics it uses, the learning model, and how the learned model is used for workflow planning. In section 5, we provide detailed experimental results and evaluate our approach in different settings. Finally, we conclude in section 6.

3. System Description

In this section, we will provide a general description of the system. We will start by defining the notations that we will use throughout the paper and then give a brief overview of the different components of the system. Its two most important components, the planner and the meta-miner will be described in subsequent sections (3.4 and 4 respectively). We will close the section by providing the formal representation of the DM workflows which we will use in the planner and the meta-miner.

Symbol	Meaning
o	a workflow operator.
e	a workflow data type.
$w_l = [o_1, \ldots, o_l]$	a ground DM workflow as a sequence of l operators.
$\mathbf{w}_{c_l} = (\mathrm{I}(p_1 \preceq_t w_l), \ldots, \mathrm{I}(p_{\lvert P\rvert} \preceq_t w_l))^{\mathrm{T}}$	the fixed length $\lvert P\rvert$-dimensional vector description of a workflow w_l
$\mathbf{x}_u = (x_1, \ldots, x_d)^{\mathrm{T}}$	the d-dimensional vector description of a dataset.
$\mathrm{r}(\mathbf{x}, \mathbf{w})$	the relative performance rank of $\mathbf{w}$ workflow on $\mathbf{x}$ dataset
g	a DM goal.
t	a HTN task.
m	a HTN method.
$\hat{O} = \{o_1, \ldots, o_n\}$	a HTN abstract operator with n possible operators.
C_l	a set of candidate workflows at some abstract operator $\hat{O}$.
S_l	a set of candidate workflows selected from C_l.

Table 1: Summary of notations used.

3.1 Notations

We will provide the most basic notations here and subsequently introduce additional notations as they are needed. We will use the term *DM experiment* to designate the execution of a DM workflow $\mathbf{w} \in \mathcal{W}$ on a dataset $\mathbf{x} \in \mathcal{X}$. We will describe a dataset $\mathbf{x}$ by a d-dimensional column vector $\mathbf{x}_u = (x_1, \ldots x_d)^{\mathrm{T}}$; we describe in detail in section 4.1.1 the dataset descriptions that we use. Each experiment will be characterized by some performance measure; for example if the mining problem we are addressing is a classification problem one such performance measure can be the accuracy. From the performance measures of the workflows applied on a given dataset $\mathbf{x}$ we will extract the relative performance rank $\mathrm{r}(\mathbf{x}, \mathbf{w}) \in \mathbb{R}_+$ of each workflow $\mathbf{w}$ for the $\mathbf{x}$ dataset. We will do so by statistically comparing the performance differences of the different workflows for the given dataset (more on that in section 4.1.3). The matrix $\mathbf{X} : n \times d$ contains the descriptions of n datasets that will be used for the training of the system. For a given workflow $\mathbf{w}$ we will have two representations. w_l will denote the l-length operator sequence that constitutes the workflow. Note that different workflows can have different lengths, so this is not a fixed length representation. $\mathbf{w}_{c_l}$ will denote the fixed-length $\lvert P\rvert$-dimensional binary vector representation of the $\mathbf{w}$ workflow; each feature of $\mathbf{w}_{c_l}$ indicates the presence or absence of some *relational* feature/pattern in the workflow. Essentially $\mathbf{w}_{c_l}$ is the propositional representation of the workflow; we will describe in more detail in section 4.1.2 how we extract this propositional representation. Finally W denotes a collection of m workflows, which will be used in the training of the system, and $\mathbf{W}$ is the corresponding $m \times \lvert P\rvert$ matrix that contains their propositional representations. Depending on the context the different workflow notations can be used interchangeably. Table 1 summarizes the most important notations.

3.2 System Architecture and Operational Pipeline

We provide in Figure 1 a high level architectural description of our system. The three blue shaded boxes are: (i) the *Data Mining Experiment Repository* (DMER) which stores all the

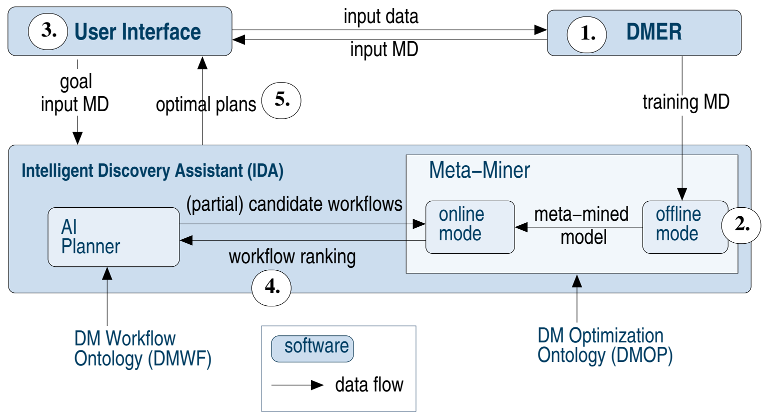

Figure 1: The meta-mining system's components and its pipeline.

base-level resources, i.e. training datasets, workflows, as well as the performance results of the application of the latter on the former; essentially DMER contains the training data that will be used to derive the models that will be necessary for the workflow planning and design; (ii) the user interface through which the user interacts with the system, specifying data mining tasks and input datasets and (iii) the *Intelligent Discovery Assistant* (IDA) which is the component that actually plans data mining workflows that will optimize some performance measure for a given dataset and data mining task that the user has provided as input. IDA constitutes the core of the system and contains a planning component and a meta-mining component which interact closely in order to deliver optimal workflows for the given input problem; we will describe these two components in detail in sections 3.4 and 4 respectively. The system operates in two modes, an offline and an online mode. In the offline mode the meta-mining component analyses past base-level data mining experiments, which are stored in the DMER, to learn a meta-mining model that associates dataset and workflow characteristics in view of performance optimization. In the online mode the meta-miner interacts with the planner to guide the planning of the workflows using the meta-mining model.

We will now go briefly through the different steps of the system's life cycle. We first need to collect in the DMER a sufficient number of base-level data mining experiments, i.e. applications of different data mining workflows on different datasets (step 1). These experiments will be used in step 2 by the META-MINER to generate a meta-mining model. More precisely we extract from the base level experiments a number of characteristics that describe the datasets and the workflows and the performance that the latter achieved when applied on the former. From these meta-data the META-MINER learns associations of dataset and workflow characteristics that lead to high performance; it does so by learning a heterogeneous similarity measure which outputs a high "similarity" of a workflow to a dataset if the former is expected to achieve a high performance when it will be applied to the latter (more details in section 4.2).

Once the model is learned, the offline phase is completed and the online phase can start. In the online phase the system directly interacts with its user. A given user can select a data mining task $g \in \mathcal{G}$, such as classification, regression, clustering, etc, as well as an input dataset on which the task should be applied; he or she might also specify the number of top-k optimal workflows that should be planned (step 3). Given a new task and an input dataset the action is transfered in the IDA where the AI-PLANNER starts the planning process. At each step of the planning process the AI-PLANNER generates a list of valid actions, partial candidate workflows, which it passes for ranking to the META-MINER according to their expected performance on the given dataset, step 4. The META-MINER ranks them using the learned meta-mining model and the planning continues with the top-ranked partial candidate workflows until the data mining task is resolved. At the end of this planning process, the top-k workflows are presented to the user in the order given by their expected performance on the input dataset. This is a greedy planning approach where at each step we select the top-k current solutions. In principle we can let the AI-PLANNER first generate all possible workflows and then have the META-MINER rank them. Then the resulting plans would not be ranked according to a local greedy approach, but they would be ranked globally and thus optimally with respect to the meta-mining model. However in general this is not feasible due to the complexity of the planning process and the combinatorial explosion in the number of plans.

3.3 DM Workflow Representation

We will now give a formal definition of a DM workflow and how we represent it. DM workflows are directed acyclic typed graphs (DAGs), in which nodes correspond to operators and edges between nodes to data input/output objects. In fact they are hierarchical DAGs since they can have *dominating* nodes/operators that contain sub-workflows. A typical example is the cross-validation operator whose control flow is given by the parallel execution of training sub-workflows, or a complex operator such as boosting. More formally, let:

- O be the set of all available operators that can appear in a DM workflow, e.g. classification operators, such as *J48*, *SVMs*, etc. O also includes *dominating* operators which are defined by one or more sub-workflows they dominate. An operator $o \in O$ is defined by its name through a labelling function $\lambda(o)$, the data types $e \in E$ of its inputs and outputs, and its direct sub-workflows if o is a *dominating* operator.

- E be the set of all available data types that can appear in a DM workflow, namely the data types of the various I/O objects that can appear in a DM workflow such as models, datasets, attributes, etc.

The graph structure of a DM workflow is a pair (O', E'), which also contains all sub-workflows if any. $O' \subset O$ is the set of vertices which correspond to all the operators used in this DM workflow and its sub-workflow(s), and $E' \subset E$ is the set of pairs of nodes, (o_i, o_j), directed edges, that correspond to the data types of the output/input objects, that are passed from operator o_i to operator o_j. Following this graph structure, the *topological sort* of a DM workflow is a permutation of the vertices of the graph such that an edge (o_i, o_j) implies that o_i appears before o_j, i.e. this is a complete ordering of the nodes of a directed acyclic graph which is given by the node sequence:

$$w_l = [o_1, .., o_l] \tag{1}$$

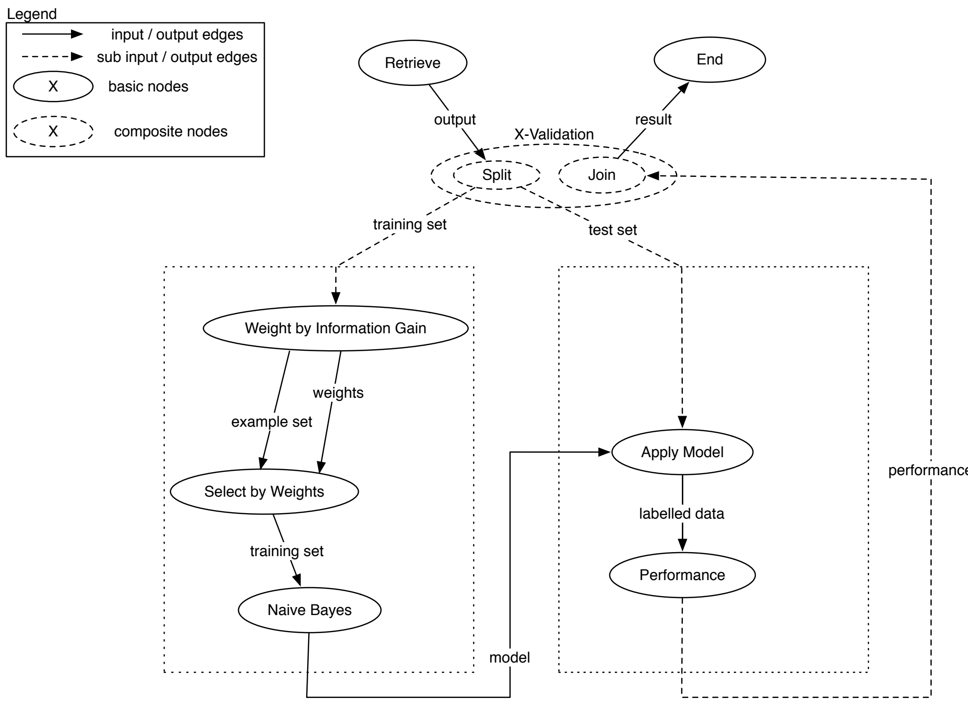

Figure 2: Example of a DM workflow that does performance estimation of a combination of feature selection and classification algorithms.

where the subscript in w_l denotes the length l (i.e. number of operators) of the topological sort. The topological sort of a DM workflow can be structurally represented with a rooted, labelled and ordered tree (Bringmann, 2004; Zaki, 2005), by doing a depth-first search over its graph structure where the maximum depth is given by expanding recursively the sub-workflows of the dominating operators. Thus the topological sort of a workflow or its tree representation is a reduction of the original directed acyclic graph where the nodes and edges have been fully ordered.

An example of the hierarchical DAG representing a RapidMiner DM workflow is given in Figure 2. The graph corresponds to a DM workflow that cross-validates a feature selection method followed by a classification model building step with the *NaiveBayes* classifier. *X-Validation* is a typical example of a dominating operator which itself is a workflow – it has two basic blocks, a training block which can be any arbitrary workflow that receives as input a dataset and outputs a model, and a testing block which receives as input a model and a dataset and outputs a performance measure. In particular, we have a training sub-workflow, in which feature weights are computed by the *InformationGain* operator, after which a number of features is selected by the *SelectByWeights* operator, followed by the final model building by the *NaiveBayes* operator. In the testing block, we have a typical sub-workflow which consists of the application of the learned model on the testing set with the *ApplyModel* operator, followed by a performance estimation given by the *Performance* operator. The topological sort of this graph is given by the ordered tree given in Figure 3.

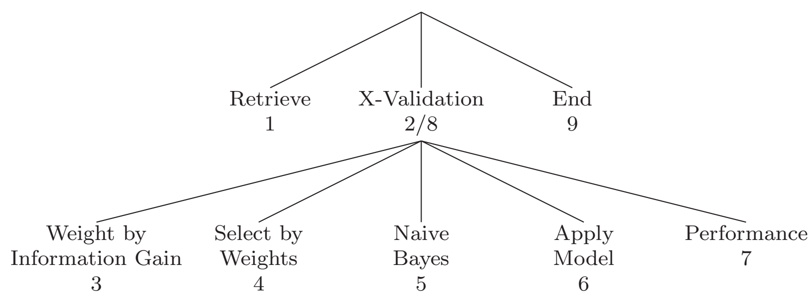

Figure 3: The topological order of the DM workflow given in Figure 2.

3.4 Workflow Planning

The workflow planner we use is based on the work of Kietz et al. (2009, 2012), and designs DM workflows using a hierarchical task network (HTN) decomposition of the CRISP-DM process model (Chapman et al., 2000). However this planner only does workflow enumeration generating all possible plans, i.e. it does not consider the expected workflow performance during the planning process and does not scale to large set of operators which explode the search space. In order to address these limitations, we presented some preliminary results in which the planner was coupled with a frequent pattern meta-mining algorithm and a nearest-neighbor algorithm to rank the partial workflows at each step of the workflow planning (Nguyen et al., 2011, 2012a). This system was also deployed by Kietz et al. (2012). The approach we followed was to prioritize the partial workflows according to the support that the frequent patterns they contained achieved on a set of workflows that performed well on a set of datasets that were similar to the input dataset for which we want to plan the workflow. However we were not learning associations between dataset and workflow characteristics, which is the approach we follow here.

In section 3.4.1 we briefly describe the HTN planner of Kietz et al. (2009, 2012); and in section 3.4.2 we describe how we use the prediction of the expected performance of a partial-workflow applied on a given dataset to guide the HTN planner. We will give the complete description of our meta-mining module that learns associations of dataset and workflow characteristics that are expected to achieve high performance in section 4.

3.4.1 HTN Planning

Given some goal $g \in \mathcal{G}$, the AI-planner will decompose in a top-down manner this goal into elements of two sets, tasks T and methods M. For each task $t \in T$ that can achieve g, there is a subset $M' \subset M$ of associated methods that share the same data input/output (I/O) object specification with t and that can address it. In turn, each method $m \in M'$ defines a sequence of operators, and/or abstract operators (see below), and/or sub-tasks, which executed in that order can achieve m. By recursively expanding tasks, methods and operators for the given goal g, the AI-planner will sequentially construct an HTN plan in which terminal nodes will correspond to operators, and non-terminal nodes to HTN task or method decompositions, or to dominating operators (*X-Validation* for instance will dominate a training and a testing sub-workflows). An example of an HTN plan is given

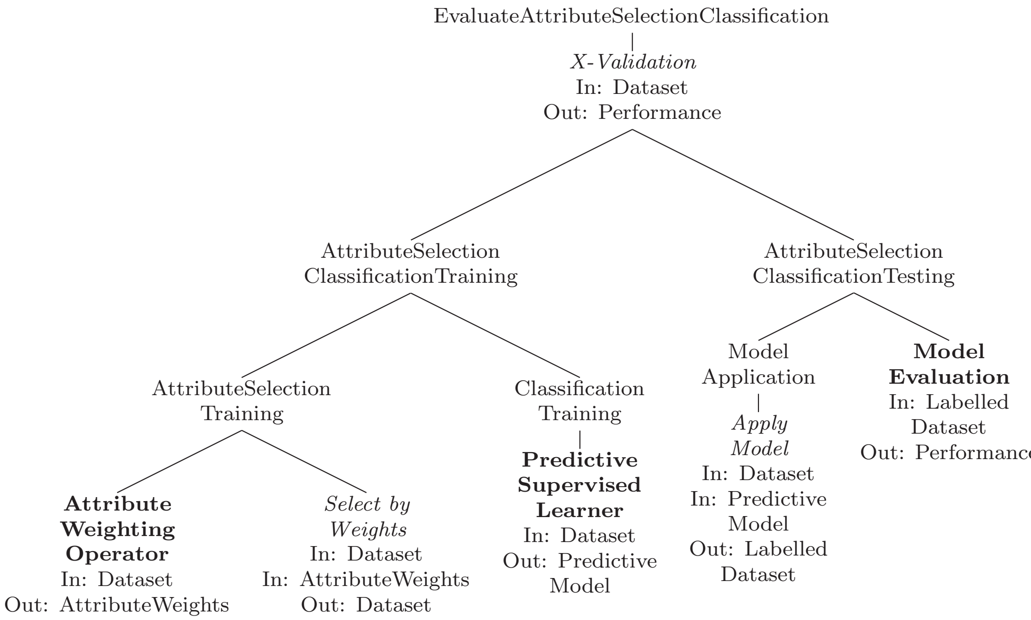

Figure 4: The HTN plan of the DM workflow given in Figure 2. Non-terminal nodes are HTN tasks/methods, except for the *dominating* operator *X-Validation*. *Abstract* operators are in bold and simple operators in italic, each of which is annotated with its I/O specification.

in Figure 4. This plan corresponds to the feature selection and classification workflow of Figure 2 with exactly the same topological sort (of operators) given in Figure 3.

The sets of goals $\mathcal{G}$, tasks T, methods M, and operators O, and their relations, are described in the DMWF ontology (Kietz et al., 2009). There, methods and operators are annotated with their pre- and post-conditions so that they can be used by the AI-planner. Additionally, the set of operators O has been enriched with a shallow taxonomic view in which operators that share the same I/O object specification are grouped under a common ancestor:

$$\hat{O} = \{o_1, \ldots, o_n\} \tag{2}$$

where $\hat{O}$ defines an *abstract* operator, i.e. an operator choice point or alternative among a set of n syntactically similar operators. For example, the abstract *AttributeWeighting* operator given in Figure 4 will include any feature weighting algorithms such as *InformationGain* or *ReliefF*, and similarly the abstract *Predictive Supervised Learner* operator will contain any classification algorithms such as *NaiveBayes* or a linear *SVM*.

The HTN planner can also plan operators that can be applied on each attribute, e.g. a continuous attribute normalization operator, or a discretization operator. It uses cyclic planning structures to apply them on subsets of attributes. In our case we use its attribute-grouping functionality which does not require the use of cyclic planning structures. More precisely if such an operator is selected for application it is applied on all appropriate attributes. Overall the HTN grammar contains descriptions of 16 different tasks and more than 100 operators, over which the planner can plan. These numbers are only limited by the modeling effort required to describe the tasks and the operators, they are not an inherent limitation of the HTN grammar/planner. Kietz et al. (2012) have developed a module for the open-source ontology editor Protege called E-ProPlan[2] to facilitate the modelling of new operators and describe task-method decomposition grammars for DM problems in the dmwf ontology.

3.4.2 The Workflow Selection Task

The AI-planner designs during the construction of an HTN plan several partial workflows, each of which will be derived by substituting an abstract operator $\hat{O}$ with one of its n operators. The number of planned workflows can be in the order of several thousands which will leave the user at a loss as to which workflow to choose for her/his problem; even worse, it is possible that the planning process never succeeds to find a valid plan and terminate due to the complexity of the search space.

To support the user in the selection of DM workflows, we can use a post-planning approach in which the designed workflows will be evaluated according to some evaluation measure in order to find the k ones that will be globally the best for the given mining problem. However, this global approach will be very computational expensive in the planning phase as we mentioned above. Instead, we will follow here a planning approach in which we will locally guide the AI-planner towards the design of DM workflows that will be optimized for the given problem, avoiding thus the need to explore the whole planning search space. Clearly, by following a local approach the cost that one has to pay is a potential reduction in performance since not all workflows are explored and evaluated, potentially missing good ones due to the greedy nature of the plan construction.

We adopt a heuristic hill climbing approach to guide the planner. At each abstract operator $\hat{O}$ we need to determine which of the n candidate operators $o \in \hat{O}$ are expected to achieve the best performance on the given dataset. More formally, we will define by:

$$C_l := \{w_{lo} = [w_{l-1} \in S_{l-1} | o \in \hat{O}]\}^{k \times n} \tag{3}$$

the set of $k \times n$ partial candidate workflows of length l which are generated from the expansion of an abstract operator $\hat{O}$ by adding one of the n candidate operators $o \in \hat{O}$ to one of the k candidate workflows of length $l-1$ that constitute the set S_{l-1} of workflows selected in the previous planning step (S_0 is the empty set see below). Now let $\mathbf{x}_u$ be a vector description of the input dataset for which we want to plan workflows which address some data mining goal g and optimize some performance measure. Moreover let $\mathbf{w}_{c_{lo}}$ be a binary vector that provides a propositional representation of the w_{lo} workflow with respect to the set of generalized relational frequent workflow patterns that it contains[3]. We construct the set S_l of selected workflows at the current planning step according to:

$$S_l := \{\arg\max_{\{w_{lo} \in C_l\}} \hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}} | g)\}^k \tag{4}$$

2. Available at http://www.e-lico.eu/eproplan.html
3. We will provide a detailed description of how we extract the $\mathbf{w}_{c_{lo}}$ descriptors in section 4.1.2, for the moment we note that these propositional representations are fixed length representations, that do not depend on l.

where $\hat{r}(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$ is the estimated performance of a workflow with the propositional description $\mathbf{w}_{c_{l_o}}$ when applied to a dataset with description $\mathbf{x}_u$, i.e. at each planning step we select the best current partial workflows according to their estimated expected performance. It is the meta-mining model, that we learn over past experiments, that delivers the estimations of the expected performance. In section 4.2 we describe how we derive the meta-mining models and how we use them to get the estimates of the expected performance.

We should stress here that in producing the performance estimate $\hat{r}(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$ the meta-mining model uses the workflow description $\mathbf{w}_{c_{l_o}}$, and not just the candidate operator descriptions. These pattern-based descriptions capture dependencies and interactions between the different operators of the workflows (again more on the $\mathbf{w}_{c_{l_o}}$ representation in section 4.1.2). This is a rather crucial point since it is a well known fact that when we construct data mining workflows we need to consider the relations of the biases of the different algorithms that we use within them, some bias combinations are better than others. The pattern based descriptions that we will use provide precisely that type of information, i.e. information on the operator combinations appearing within the workflows.

In the next section we will provide the complete description of the meta-miner module, including how we characterize datasets, workflows and the performance of the latter applied to the former, and of course how we learn the meta-mining models and use them in planning.

4. The Meta-Miner

The meta-miner component operates in two modes. In the offline mode it learns from past experimental data a meta-mining model which provides the expected performance estimates $\hat{r}(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$. In the on-line mode it interacts with the AI-planner at each step of the planning process, delivering the $\hat{r}(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$ estimates which are used by the planner to select workflows at each planning step according to Eq.(4).

The rest of the section is organised as follows. In subsection 4.1.1 we explain how we describe the datasets, i.e. how we derive the $\mathbf{x}_u$ dataset descriptors; in subsection 4.1.2 how we derive the propositional representation $\mathbf{w}_{c_{l_o}}$ of the data mining workflows and in subsection 4.1.3 how we rank the workflows according to their performance on a given dataset, i.e. $r(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$. Finally in subsection 4.2 we explain how we build models from past mining experiments which will provide the expected performance estimations $\hat{r}(\mathbf{x}_u, \mathbf{w}_{c_{l_o}}|g)$ and how we use these models within the planner.

4.1 Meta-Data and Performance Measures

In this section we will describe the meta-data, namely dataset and workflow descriptions, and the performance measures that are used by the meta-miner to learn meta-mining models that associate dataset and workflow characteristics in view of planning DM workflows that optimize a performance measure.

4.1.1 Dataset Characteristics

The idea to characterize datasets has been a full-fledged research problem from the early inception of meta-learning until now (Michie, Spiegelhalter, Taylor, & Campbell, 1994; Köpf, Taylor, & Keller, 2000; Pfahringer, Bensusan, & Giraud-Carrier., 2000; Soares &

Brazdil, 2000; Hilario & Kalousis, 2001; Peng, Flach, Soares, & Brazdil, 2002; Kalousis, Gama, & Hilario, 2004). Following state-of-art dataset characteristics, we will characterize a dataset $\mathbf{x} \in \mathcal{X}$ by the three following groups of characteristics.

- *Statistical and information-theoretic measures*: this group refers to data characteristics defined in the STATLOG (Michie et al., 1994; King, Feng, & Sutherland, 1995) and METAL projects[4] (Soares & Brazdil, 2000), and it includes *number of instances, number of classes, proportion of missing values, proportion of continuous / categorical features, noise signal ratio, class entropy, mutual information.* They mainly describe attribute statistics and class distributions of a given dataset sample.

- *Geometrical and topological measures*: this group concerns new measures which try to capture geometrical and topological complexity of class boundaries (Ho & Basu, 2002, 2006), and it includes *non-linearity, volume of overlap region, maximum Fisher's discriminant ratio, fraction of instance on class boundary, ratio of average intra/inter class nearest neighbour distance.*

- *Landmarking and model-based measures*: this group is related to measures asserted with fast machine learning algorithms, so called *landmarkers* (Pfahringer et al., 2000), and its derivative based on the learned models (Peng et al., 2002), and it includes error rates and pairwise $1-p$ values obtained by landmarkers such as *1NN* or *DecisionStump*, and histogram weights learned by *Relief* or *SVM*. We have extended this last group with new landmarking methods based on the weight distribution of feature weighting algorithms such as *Relief* or *SVM* where we have build twenty different histogram representations of the discretized feature weights.

Overall, our system makes use of a total of $d = 150$ numeric characteristics to describe a dataset. We will denote this vectorial representation of a dataset $\mathbf{x} \in \mathcal{X}$ by $\mathbf{x}_u$. We have been far from exhaustive in the dataset characteristics that we used, not including characteristics such as subsampling landmarks (Leite & Brazdil, 2010). Our main goal in this work is not to produce a comprehensive set of dataset descriptors but to design a DM workflow planning system that given a set of dataset characteristics, coupled with workflow descriptors, can plan DM workflows that optimize some performance measure.

4.1.2 Workflow Characteristics

As we have seen in section 3.3 workflows are graph structures that can be quite complex containing several nested sub-structures. These are very often difficult to analyze not only because of their "spaghetti-like" structure but also because we do not have any information on which subtask is addressed by which workflow component (Van der Aalst & Giinther, 2007). Process mining addresses this problem by mining for generalized patterns over workflow structures (Bose & der Aalst, 2009).

To characterize DM workflows we will follow a process mining like approach; we will extract generalized, relational, frequent patterns over their tree representations, that we will use to derive propositional representations of them. The possible generalizations will

4. http://www.metal-kdd.org/

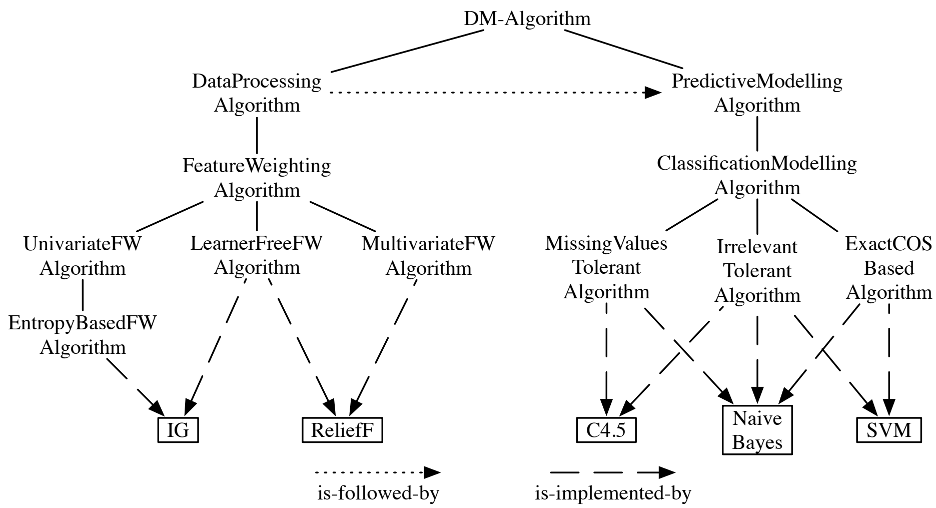

Figure 5: A part of the DMOP's algorithm taxonomies. Short dashed arrows represent the `is-followed-by` relation between DM algorithms, and long dashed arrows represent the `is-implemented-by` relation between DM operators and DM algorithms.

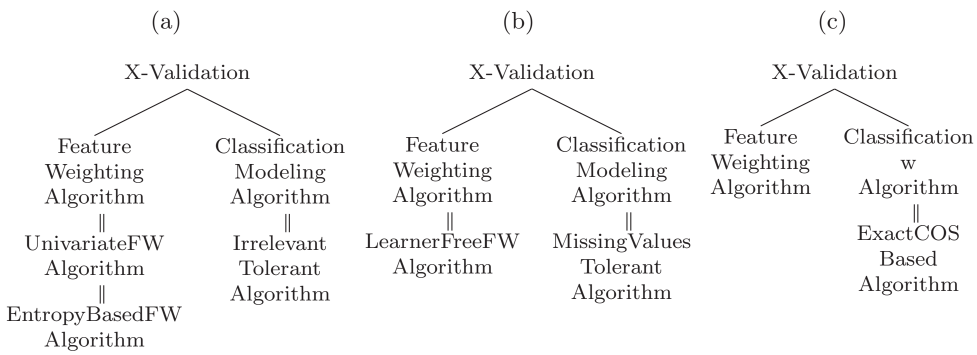

Figure 6: Three workflow patterns with cross-level concepts. Thin edges depict workflow decomposition; double lines depict DMOP's concept subsumption.

be described by domain knowledge which, among other knowledge, will be given by a data mining ontology. We will use the *Data Mining Optimization* (DMOP) ontology (Hilario et al., 2009, 2011). Briefly, this ontology provides a formal conceptualization of the DM domain by describing DM algorithms and defining their relations in terms of DM tasks, models and workflows. It describes learning algorithms such as *C4.5*, *NaiveBayes* or *SVM*, according to their learning behavior such as their bias/variance profile, their sensitivity to the type of attributes, etc. For instance, the algorithms cited above are all tolerant to irrelevant attributes, but only *C4.5* and *NaiveBayes* algorithms are tolerant to missing values, whereas only the *SVM* and *NaiveBayes* algorithms have an exact cost function. Algorithm characteristics and families are classified as taxonomies in DMOP under the primitive concept of DM-Algorithm. Moreover, DMOP specifies workflow relations, algorithm order, with the is-followed-by relation and relates workflow operators with DM algorithms with the is-implemented-by relation. Figure 5 shows a snapshot of the DMOP's algorithm taxonomies with ground operators at the bottom related to the DM algorithms they implement.

To mine generalized relational patterns from DM workflows, we will follow the method presented by Hilario et al. (2011). First, we use the DMOP ontology to annotate a set W of workflows. Then, we extract from this set generalized patterns using a frequent pattern mining algorithm. Concretely, for each operator contained in the parse tree of a training DM workflow $w_l \in W$, we insert into the tree branch above the operator the taxonomic concepts, ordered from top to bottom, that are implemented by this operator, as these are given in the DMOP. The result is a new parse tree that has additional nodes which are DMOP's concepts. We will call this parse tree an augmented parse tree. We then reorder the nodes of each augmented parse tree to to satisfy DMOP's algorithm taxonomies and relations. For example a feature selection algorithm is typically composed of a feature weighting algorithm followed by a decision rule that selects features according to some heuristics. The result is a set of augmented and reordered workflow parse trees. Over this representation, we apply a tree mining algorithm (Zaki, 2005) to extracts a set P of frequent patterns. Each pattern corresponds to a tree that appears frequently within the augmented parse trees; we mine patterns that have a support higher or equal to five. In principle we can go as low as a support of one, exploding the dimensionality of the description of workflows, with what probably will be features of poor discriminatory power. Nevertheless since our meta-mining models rely on metric learning, which is able to learn the importance of the different meta-features, they would be able to cope also with such a scenario. Note that to extract the workflow characteristics we could have used other different techniques such as graph mining directly over the graph structures defined by the workflows and the ontology, the main reason for not doing that was the computational cost of the latter approaches, as well as the fact that frequent pattern mining propositionalization is known to work very well.

In Figure 6 we give examples of the mined patterns. Note that the extracted patterns are generalized, in the sense that they contain entities defined at different abstraction levels, as these are provided by the DMOP ontology. They are relational because they describe relations, such as order relations, between the structures that appear within a workflow, and they also contain properties of entities as these are described in the DMOP ontology. For example pattern (c) of Figure 6 states that we have a feature weighting algorithm (abstract concept) followed by (relation) a classification algorithm that has an exact cost function (property), within a cross-validation.

We use the set P of frequent workflow patterns to describe any DM workflow $w_l \in \mathcal{W}$ through the patterns $p \in P$ that this w_l workflow contains. The propositional description of a workflow w_l is given by the $|P|$-length binary vector:

$$\mathbf{w}_{c_l} = (\mathrm{I}(p_1 \preceq_t w_l), \ldots, \mathrm{I}(p_{|P|} \preceq_t w_l))^{\mathrm{T}} \in \{0,1\}^{|P|} \tag{5}$$

where $\preceq_t$ denotes the induced tree relation (Zaki, 2005) and $\mathrm{I}(p_i \preceq_t w_l)$ returns one if the frequent pattern, p_i, appears within the workflow and zero otherwise.

We will use this propositional workflow representation together with a tabular representation of the datasets characteristics to learn the meta-mining models which we will describe in the next section. Although we could have used tree or even graph properties to represent workflows, propositionalization is a standard approach used extensively and successfully in learning problems in which the learning instances are complex structures (Kramer, Lavrač, & Flach, 2000).

Also, the propositional workflow representation can easily deal with the parameter values of the different operators that appear within the workflows. To do so, we can discretize the range of values of a continuous parameter to ranges such as low, medium, high, or other ranges depending on the nature of the parameter, and treat these discretized values as simply a property of the operators. The resulting patterns will now be "parameter-aware"; they will include information on the parameter range of the mined operators and they can be used to support also the parameter setting during the planning of the DM workflows. However within this paper we will not explore this possibility.

4.1.3 Performance-Based Ranking of DM Workflow

To characterize the performance of a number of workflows applied on a given dataset we will use a relative performance rank schema that we will derive using statistical significance tests. Given the estimations of some performance measure of the different workflows on a given dataset we use a statistical significance test to compare the estimated performances of every pair of workflows. If within a given pair one of the workflows was significantly better than the other then it gets one point and the other gets zero points. If there was no significance difference then both workflows get half a point. The final performance rank of a workflow for the given dataset is simply the sum of its points over all the pairwise performance comparisons, the higher the better. We will denote this relative performance rank of a workflow $\mathbf{w}_c$ applied on dataset $\mathbf{x}_u$ by $\mathrm{r}(\mathbf{x}_u, \mathbf{w}_c)$. Note that if a workflow was not applicable, or not executed, on the dataset $\mathbf{x}$, we set its rank score to the default value of zero which means that the workflow is not appropriate (if not yet executed) for the given dataset. When the planning goal g is the classification task, we will use as evaluation measure in our experiments the classification accuracy, estimated by ten-fold cross-validation, and do the significance testing using McNemar's test, with a significance level of 0.05.

In the next section we will describe how we build the meta-mining models from the past data-mining experiments using the meta-data and the performance measures we have described so far and how we use these models to support the DM workflow planning.

4.2 Learning Meta-mining Models for Workflow Planning

Before starting to describe in detail how we build the meta-mining models let us take a step back and give a more abstract picture of the type of meta-mining setting that we will address. In the previous sections, we described two types of learning instances: datasets $\mathbf{x} \in \mathcal{X}$ and workflows $\mathbf{w} \in \mathcal{W}$. Given the set of datasets and the set of workflows stored in the DMER, the META-MINER will build from these, two training matrices $\mathbf{X}$ and $\mathbf{W}$. The $\mathbf{X} : n \times d$ dataset matrix, has as its i^{th} row the description $\mathbf{x}_{u_i}$ of the ith dataset. The $\mathbf{W} : m \times |P|$ workflow matrix, has as its j^{th} row the description $\mathbf{w}_{c_j}$ of the jth workflow. We also have a preference matrix $\mathbf{R} : n \times m$, where $R_{ij} = \mathrm{r}(\mathbf{x}_{u_i}, \mathbf{w}_{c_j})$, i.e. it gives the relative performance rank of the workflow $\mathbf{w}_j$ when applied to the dataset $\mathbf{x}_i$ with respect to the other workflows. We can see R_{ij} as a measure of the appropriateness or match of the $\mathbf{w}_j$ workflow for the $\mathbf{x}_i$ dataset. The ith line of the $\mathbf{R}$ matrix contains the vector of the relative performance ranks of the workflows that were applied on the $\mathbf{x}_{u_i}$ dataset. The meta-miner will take as input the $\mathbf{X}, \mathbf{W}$ and $\mathbf{R}$ matrices and will output a model that predicts the expected performance, $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_c)$, of a workflow $\mathbf{w}$ applied to a dataset $\mathbf{x}$.

We construct the meta-mining model using similarity learning, exploiting two basic strategies initially presented in the context of DM workflow selection (Nguyen et al., 2012b). Here we will give only a high level presentation of them, for more details the interested user should refer to the original paper. In the first strategy we learn homogeneous similarity measures, measuring similarity of datasets and similarity of workflows, which we then use to derive the $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_c|g)$ estimates. In the second we learn heterogeneous similarity measures which directly estimate the appropriateness of a workflow for a dataset, i.e. they produce direct estimates of $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_c|g)$.

4.2.1 Learning Homogeneous Similarity Measures

Our goal is to provide meta-mining models that are good predictors of the performance of a workflow applied to a dataset. In the simplest approach we want to learn a good similarity measure on the dataset space that will deem two datasets to be similar if a set of workflows applied to both of them will result in a similar relative performance, i.e. if we order the workflows according to the performance they achieve on each dataset then two datasets will be similar if the workflow orders are similar. Thus the learned similarity measure on the dataset space should be a good predictor of the similarity of the datasets as this is determined by the relative performance order of the workflows. In a completely symmetrical manner we will consider two workflows to be similar if they achieve similar relative performance scores on a set of datasets. Thus in the case of workflows we will learn a similarity measure on the workflow space that is a good predictor of the similarity of their relative performance scores over a set of datasets.

Briefly, we learn two Mahalanobis metric matrices, $\mathbf{M}_{\mathcal{X}}, \mathbf{M}_{\mathcal{W}}$, over the datasets and the workflows respectively, by optimizing the two following convex metric learning optimization problems:

$$\begin{aligned} \min_{\mathbf{M}_{\mathcal{X}}} \quad & F_1 = ||\mathbf{R}\mathbf{R}^{\mathrm{T}} - \mathbf{X}\mathbf{M}_{\mathcal{X}}\mathbf{X}^{\mathrm{T}}||_F^2 + \mu\,\mathrm{tr}(\mathbf{M}_{\mathcal{X}}) \\ s.t. \quad & \mathbf{M}_{\mathcal{X}} \succeq 0 \end{aligned} \tag{6}$$

and

$$\begin{aligned} \min_{\mathbf{M}_{\mathcal{W}}} \quad & F_2 = ||\mathbf{R}^{\mathrm{T}}\mathbf{R} - \mathbf{W}\mathbf{M}_{\mathcal{W}}\mathbf{W}^{\mathrm{T}}||_F^2 + \mu\,\mathrm{tr}(\mathbf{M}_{\mathcal{W}}) \\ s.t. \quad & \mathbf{M}_{\mathcal{W}} \succeq 0 \end{aligned} \tag{7}$$

where $||.||_F$ is the Frobenius matrix norm, $\mathrm{tr}(\cdot)$ the matrix trace, and $\mu \geq 0$ is a parameter controlling the trade-off between empirical error and the metric complexity to control over-fitting. The $\mathbf{R}\mathbf{R}^{\mathrm{T}} : n \times n$ matrix reflects the similarity of the relative workflow performance vectors over the different dataset pairs which the learned dataset similarity metric should reflect. The $\mathbf{R}^{\mathrm{T}}\mathbf{R} : m \times m$ matrix gives the respective similarities for the workflows. For more details on the learning problem and how we solve it, see the work of Nguyen et al. (2012b).

Note that so far we do not have a model that computes the expected relative performance $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}})$. In the case of the homogeneous metric learning we will compute it in the on-line mode during the planning phase; we will describe right away how we do so in the following paragraph.

Planning with the homogeneous similarity metrics (P1) We will use the two learned Mahalanobis matrices, $\mathbf{M}_{\mathcal{X}}$, $\mathbf{M}_{\mathcal{W}}$, to compute the dataset similarity and the workflow similarity out of which we will finally compute the estimates $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}})$ at each planning step.

Concretely, prior to planning we determine the similarity of the input dataset $\mathbf{x}_u$ (for which we want to plan optimal DM workflows) to each of the training datasets $\mathbf{x}_{u_i} \in \mathbf{X}$ using the $\mathbf{M}_{\mathcal{X}}$ dataset metric to measure the dataset similarities. The Mahalanobis similarity of two datasets, $\mathbf{x}_u, \mathbf{x}_{u_i}$, is given by

$$s_{\mathcal{X}}(\mathbf{x}_u, \mathbf{x}_{u_i}) = \mathbf{x}_u^T \mathbf{M}_{\mathcal{X}} \mathbf{x}_{u_i} \tag{8}$$

Then, during planning at each planning step we determine the similarity of each candidate workflow $w_{lo} \in C_l$ to each of the training workflows $\mathbf{w}_{c_j}$ of $\mathbf{W}$, by

$$s_{\mathcal{W}}(\mathbf{w}_{c_{lo}}, \mathbf{w}_{c_j}) = \mathbf{w}_{c_{lo}}^T \mathbf{M}_{\mathcal{W}} \mathbf{w}_{c_j}. \tag{9}$$

Finally we derive the $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}})$ estimate through a weighted average of the elements of the $\mathbf{R}$ matrix. The weights are given by the similarity of the input dataset $\mathbf{x}_u$ to the training datasets, and the similarities of the candidate workflow $\mathbf{w}_{c_{lo}}$ to each of the training workflows. More formally the expected rank is given by:

$$\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}}|g) = \frac{\sum_{\mathbf{x}_{u_i} \in \mathbf{X}} \sum_{\mathbf{w}_{c_j} \in \mathbf{W}} \omega_{\mathbf{x}_{u_i}} \, \omega_{\mathbf{w}_{c_j}} \, \mathrm{r}(\mathbf{x}_{u_i}, \mathbf{w}_{c_j}|g)}{\sum_{\mathbf{x}_{u_i} \in \mathbf{X}} \sum_{\mathbf{w}_{c_j} \in \mathbf{W}} \omega_{\mathbf{x}_{u_i}} \, \omega_{\mathbf{w}_{c_j}}} \tag{10}$$

$\omega_{\mathbf{x}_{u_i}}$ and $\omega_{\mathbf{w}_{c_j}}$ are the Gaussian weights given by $\omega_{\mathbf{x}_{u_i}} = \exp(s_{\mathcal{X}}(\mathbf{x}_u, \mathbf{x}_{u_i})/\tau_x)$ and $\omega_{\mathbf{w}_{c_j}} = \exp(s_{\mathcal{W}}(\mathbf{w}_{c_{lo}}, \mathbf{w}_{c_j})/\tau_w)$; τ_x and τ_w are the kernel widths that control the size of the neighbors in the data and workflow spaces respectively (Smart & Kaelbling, 2000; Forbes & Andre, 2000).

Using the rank performance estimates delivered by Eq.(10), we can select at each planning step the best candidate workflows set, S_l, according to Eq.(4). We will call the resulting planning strategy P1. Under P1 the expected performance of the set of selected candidate workflows S_l greedily increases until we deliver the k DM complete workflows which are expected to achieve the best performance on the given dataset.

4.2.2 Learning a Heterogeneous Similarity Measure

The P1 planning strategy makes use of two similarity measures that are learned independently of each other, each one defined in its own feature space. This is a simplistic assumption because it does not model for the interactions of workflows and datasets, we know that certain types of DM workflows are more appropriate for datasets with certain types of characteristics. In order to address this limitation, we will define a heterogeneous metric learning problem in which we will directly estimate the similarity/appropriateness of a workflow for a given dataset as this is given by the $\mathrm{r}(\mathbf{x}_u, \mathbf{w}_c)$ relative performance measure.

Since learning a Mahalanobis metric is equivalent to learning a linear transformation we can rewrite the two Mahalanobis metric matrices described previously as $\mathbf{M}_{\mathcal{X}} = \mathbf{U}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{M}_{\mathcal{W}} = \mathbf{V}\mathbf{V}^{\mathrm{T}}$. $\mathbf{U} : d \times t$ and $\mathbf{V} : |P| \times t$ are the respective linear transformation matrices with dimensionality $t = \min(\mathrm{rank}(\mathbf{X}), \mathrm{rank}(\mathbf{W}))$. To learn a heterogeneous similarity measure between datasets and workflows using these two linear transformations we solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} F_4 &= ||\mathbf{R} - \mathbf{X}\mathbf{U}\mathbf{V}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}||_F^2 + ||\mathbf{R}\mathbf{R}^{\mathrm{T}} - \mathbf{X}\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}||_F^2 \\ &+ ||\mathbf{R}^{\mathrm{T}}\mathbf{R} - \mathbf{W}\mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}||_F^2 + \frac{\mu}{2}(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2) \end{aligned} \tag{11}$$

using an alternating gradient descent algorithm, where we first optimize for $\mathbf{U}$ keeping $\mathbf{V}$ fixed and vice versa. The optimization problem is non-convex and the algorithm will converge to a local minimum. The first term is similar to the low-rank matrix factorization of Srebro, Rennie, and Jaakkola (2005). However the factorization that we learn here is a *function* of the dataset and workflow feature spaces and as a result it can address samples that are out of the training instances, also known as the *cold start* problem in recommendation systems. In the case of the DM workflow planning problem this is a strong requirement because we need to be able to plan workflows for datasets that have never been seen during training, and also be able to qualify workflows that have also not been seen during training. The second and third terms define metrics that reflect the performance-based similarities of datasets and workflows respectively (along the lines of the homogeneous metrics given previously), while together they give directly the similarity/appropriateness of a DM workflow for a dataset by estimating the expected relative predictive performance as:

$$\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}}|g) = \mathbf{x}_u\mathbf{U}\mathbf{V}^T\mathbf{w}_{c_{lo}}^T \tag{12}$$

We can see the heterogeneous similarity metric as performing a projection of the dataset and workflow spaces on a common latent space on which we can compute a standard similarity between the projections. Again for more details, see the work of Nguyen et al. (2012b).

Planning with the heterogeneous similarity measure (P2) Planning with the heterogeneous similarity measure, a strategy we will denote by P2, is much simpler than planning with the homogeneous similarity measures. Given an input dataset described by $\mathbf{x}_u$ at each step of the planning we make use of the relative performance estimate $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}}|g)$ delivered by Eq.(12) to select the set of best workflows S_l from the set of partial workflows C_l using the selection process described by Eq.(4). Unlike the planning strategy P1 which computes $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}}|g)$ through a weighted average with the help of the two independently learned similarity metrics, P2 relies on a heterogeneous metric that directly computes $\hat{\mathrm{r}}(\mathbf{x}_u, \mathbf{w}_{c_{lo}}|g)$, modeling thus explicitly the interactions between dataset and workflow characteristics.

We should note here that both P1 and P2 planning strategies are able to construct workflows even over pools of ground operators that include operators with which we have never experimented with in the baseline experiments, provided that these operators are well described within the dmop. This is because the meta-mining models that the planner uses to prioritize the workflows rely on the $w_{c_{l_o}}$ descriptions of a workflow which are generalized descriptions over workflows and operators.

In the next section we evaluate the ability of the two planning strategies that we have introduced to plan DM workflows that optimize the predictive performance and compare them to a number of baseline strategies under different scenarios.

5. Experimental Evaluation

We will evaluate our approach on the data mining task of classification. The reasons for that are rather practical. Classification is a supervised task which means that there is a ground truth against which we can compare the results produced by some classification algorithm, using different evaluation measures such as accuracy, error, precision etc; for other mining tasks such as clustering, performance evaluation and comparison is a bit more problematic due to the lack of ground truth. It has been extensively studied, and it is extensively used in many application fields, resulting in a plethora of benchmark datasets, which we can easily reuse to construct our base-level experiments as well as to evaluate our system. Moreover, it has been extensively addressed in the context of meta-learning, providing baseline approaches against which to compare our approach. Finally our approach requires that the different algorithms and operators that we use are well described in the dmop ontology. Due to the historical predominance of the classification task and algorithms as well as their extensive use in real world problems, we started developing dmop from them; as a result the task of classification and the corresponding algorithms are well described.

Having said all this, we should emphasize that our approach is not limited to the task of classification. It can be applied to any mining task for which we can define an evaluation measure, collect a set of benchmark datasets on which we will perform the base-level experiments, and provide descriptions of the task and the respective algorithms in the dmop.

To train and evaluate our approach, we have collected a set of benchmark classification datasets. We have applied on them a set of classification data mining workflows. From these base-level experiments we learn our meta-mining models which are then used by the planner to plan data mining workflows. Then we challenge the system with new datasets which were not used in the training of the meta-mining models, datasets for which it has to plan new classification workflows that will achieve a high level of predictive performance. We will explore two distinct evaluation scenarios. In the first one, we will constrain the system so that it plans DM workflows by selecting operators from a restricted operator pool, namely operators with which we have experimented in the base-level experiments. Thus these are operators that are characterized in the dmop ontology and are tested in the base-level experiments; we will call them *tested* operators. In the second scenario we will allow the system to also choose from operators with which we have never experimented but which are characterized in the dmop ontology; we will call these operators *untested* operators. The goal of the second scenario is to evaluate the extend to which the system can *effectively* use untested operators in the workflows it designs.

Type	Operator	Abbr.	Parameters
FS/tested	Information Gain	*IG*	-#features selected $k = 10$
FS/tested	Chi-Square	*CHI*	-#features selected $k = 10$
FS/tested	ReliefF	*RF*	-#features selected $k = 10$
FS/tested	Recursive feature elimination with SVM	*SVMRFE*	-#features selected $k = 10$
FS/untested	Information Gain Ratio	*IGR*	-#features selected $k = 10$
CL/tested	One-nearest-neighbor	*1NN*	
CL/tested	C4.5	*C4.5*	-pruning confidence $C = 0.25$ -min. inst. per leaf $M = 2$
CL/tested	CART	*CART*	-pruning confidence $C = 0.25$ -min. inst. per leaf $M = 2$
CL/tested	NaiveBayes with normal density estimation	*NB*	
CL/tested	Logistic regression	*LR*	
CL/tested	Linear kernel SVM	SVM_l	-complexity $C = 1$
CL/tested	Gaussian kernel SVM	SVM_r	-complexity $C = 1$ -gamma $\gamma = 0.1$
CL/untested	Linear discriminant analysis	*LDA*	
CL/untested	Rule induction	*Ripper*	
CL/untested	Random decision tree	*RDT*	
CL/untested	Perceptron neural network	*NNet*	

Table 2: Table of operators we used to design DM workflows for the 65 datasets. The type corresponds to feature selection (FS) or classification (CL) operators. The operators that have been experimented are marked as "tested", otherwise "untested".

5.1 Base-Level Datasets and DM Workflows

To construct the base-level experiments, we have collected 65 real world datasets on genomic microarray or proteomic data related to cancer diagnosis or prognosis, mostly from The National Center for Biotechnology Information[5]. As is typical with such datasets, the datasets we use are characterized by high-dimensionality and small sample size, and a relatively low number of classes, most often two. They have an average of 79.26 instances, 15268.57 attributes, and 2.33 classes.

To build the base-level experiments, we applied on these datasets workflows that consisted either of a single classification algorithm, or of a combination of feature selection and a classification algorithm. Although the HTN planner we use (Kietz et al., 2009, 2012) is able to generate much more complex workflows, over 16 different tasks, with more than 100 operators, we had to limit ourselves to the planning of classification, or feature selection and classification, workflows simply because the respective tasks, algorithms and operators are well annotated in the DMOP. This annotation is important for the characterization of the workflows and the construction of good meta-mining models which are used to guide the planning. Nevertheless, the system is directly usable on planning scenarios of any com-

5. http://www.ncbi.nlm.nih.gov/

plexity, as these are describe in the HTN grammar, provided that the appropriate tasks, algorithms and operators are annotated in the DMOP ontology.

We used four feature selection algorithms together with seven classification algorithms to build our set of base-level training experiments. These are given in Table 2, noted as *tested*. As we have mentioned previously, we can also plan over the operators' parameters by discretizing the range of values of the parameters and treating them as properties of the operators. Another alternative is to use inner cross-validation to automatically select over a set of parameter values; strictly speaking, in that case, we would not be selecting a standard operator but its cross-validated variant. Nevertheless, this would incur a significant computational cost.

Overall, we have seven workflows that only contained a classification algorithm, and 28 workflows that had a combination of a feature selection with a classification algorithm, resulting to a total of 35 workflows applied on 65 datasets which corresponds to 2275 base-level DM experiments. The performance measure we use is accuracy which we estimate using ten-fold cross-validation. For all algorithms, we used the implementations provided in the RapidMiner DM suite (Klinkenberg et al., 2007).

As already said, we have two evaluation settings. In the first, Scenario 1, we constrain the system to plan workflows using only the tested operators. In the second, Scenario 2, we allow the system to select also from untested operators. These additional operators are also given in Table 2, denoted as untested. The total number of possible workflows in this setting is 62.

5.2 Meta-learning & Default Methods

We will compare the performance of our system against two baseline methods and a default strategy. The two baseline methods are simple approaches that fall in the more classic meta-learning stream but instead of selecting between individual algorithms they select between workflows. Thus they cannot plan DM workflows and they can only be used in a setting in which all workflows to choose from have been seen in the model construction phase.

The first meta-learning method that we will use, which we will call *Eucl*, is the standard approach in meta-learning (Kalousis & Theoharis, 1999; Soares & Brazdil, 2000), which makes use of the Euclidean based similarity over the dataset characteristics to select the N most similar datasets to the input dataset $\mathbf{x}_u$ for which we want to select workflows and then averages their workflow rank vectors to produce the average rank vector:

$$\frac{1}{N}\sum_{i}^{N}\mathbf{r}_{\mathbf{x}_{u_i}}, \mathbf{x}_{u_i} \in \{\arg\max_{\mathbf{x}_{u_i}\in\mathbf{X}} \mathbf{x}_u^T\mathbf{x}_{u_i}\}^N \tag{13}$$

which it uses to order the different workflows. Thus this method simply ranks the workflows according to the average performance they achieve over the neighborhood of the input dataset. The second meta-learning method that we call *Metric* makes use of the learned dataset similarity measure given by Eq.(8) to select the N most similar datasets to the input dataset and then averages as well their respective workflow rank vectors:

$$\frac{1}{N}\sum_{i}^{N}\mathbf{r}_{\mathbf{x}_{u_i}}, \mathbf{x}_{u_i} \in \{\arg\max_{\mathbf{x}_{u_i}\in\mathbf{X}} \mathbf{x}_u^T\mathbf{M}_{\mathcal{X}}\mathbf{x}_{u_i}\}^N \tag{14}$$

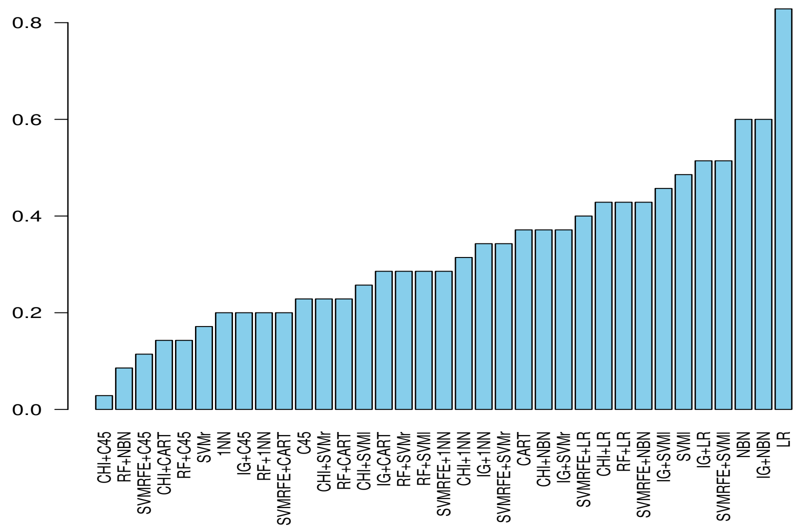

Figure 7: Percentage of times that a workflow is among the top-5 workflows over the different datasets.

For the default recommendation strategy, we will simply use the average of the $\mathbf{r}_{\mathbf{x}_{u_i}}$ workflow rank vectors over the collection of training datasets:

$$\frac{1}{n}\sum_{i}^{n}\mathbf{r}_{\mathbf{x}_{u_i}}, \forall \mathbf{x}_{u_i} \in \mathbf{X} \tag{15}$$

to rank and select the workflows. We should note that this is a rather difficult baseline to beat. To see why this is the case we plot in Figure 7 the percentage of times that each of the 35 DM workflows appears among the top-5 worfklows over the 65 datasets. The top workflow, consisting of just the *LR* algorithm, is among the top-5 for more than 80% of the datasets. The next two workflows, *NBN* and *IG* with *NBN*, are among the top-5 for almost 60% of the datasets. In other words if we select the top-5 workflows using the default strategy then in roughly 80% of the datasets *LR* will be correctly between them, while for *NBN* and *IG* with *NBN* this percentage is around 60%. Thus the set of dataset we have here is quite similar with respect to the workflows that perform better on them, making the default strategy a rather difficult one to beat.

5.3 Evaluation Methodology

To estimate the performance of the planned workflows in both evaluation scenarios we will use leave-one-dataset-out, using each time 64 datasets on which we build the meta-mining models and one dataset for which we plan.

We will evaluate each method by measuring how well the list, L, of top-k ranked workflows, that it delivers for a given dataset, correlates with the "true" list, T, of top-k ranked workflows for that dataset using a rank correlation measure. We place true between quotes because in the general case, i.e. when we do not restrict the choice of operators to a specific set, we cannot know which are the true best workflows unless we exhaustively examine an exponential number of them, however since here we select from a restricted list of operators we can have the set of the best. More precisely, to measure the rank correlation between two lists L and T, we will use the Kendall distance with p penalty, which we will denote $K^{(p)}(L,T)$ (Fagin, Kumar, & Sivakumar, 2003). The Kendall distance gives the number of exchanges needed in a bubble sort to convert one list to the other. It assigns a penalty of p to each pair of workflows such that one workflow is in one list and not in the other; we set $p = 1/2$. Because $K^{(1/2)}(L,T)$ is not normalized, we propose to define the normalized Kendall similarity *Ks(L,T)* as:

$$Ks(L,T) = 1 - \frac{K^{(\frac{1}{2})}(L,T)}{u} \tag{16}$$

and takes values in $[0,1]$. u is the upper bound of $K^{(\frac{1}{2})}(L,T)$ given by $u = 0.5k(5k+1) - 2\sum_{i=1}^{k} i$, derived from a direct application of lemma 3.2 of the work of Fagin et al. (2003), where we assume that the two lists do not share any element. We will qualify each method, m, including the two baselines, by its Kendall similarity gain, $Kg(m)$, i.e. the gain (or loss) it achieves with respect to the default strategy for a given datasets, which we compute as:

$$Kg(m)(L,T) = \frac{Ks(m)(L,T)}{Ks(def)(L,T)} - 1 \tag{17}$$

For each method, we will report its average Kendall similarity gain overall the datasets, $\overline{Kg}(m)$. Note that, in Scenario 1, the default strategy is based on the average ranks of the 35 workflows. In Scenario 1, the default strategy is based on the average ranks of the 62 workflows, which we also had to experiment with in order to set the baseline.

In addition to see how well the top-k ranked list of workflows, that a given method suggests for a given dataset, correlates to the true list, we also compute the average accuracy that the top-k workflows it suggests achieve for the given dataset, and we report the average overall datasets.

5.4 Meta-mining Model Selection

At each iteration of the leave-one-dataset-out evaluation of the planning performance, we rebuild the meta-mining model and we tune its μ parameter of the Mahalanobis metric learning using inner ten-fold cross-validation; we select the μ value that maximizes the Spearman rank correlation coefficient between the predicted workflow rank vectors and the real rank vectors. For the heterogenous metric, we used the same parameter setting defined by Nguyen et al. (2012b). For the two meta-learning methods, we fixed the number N of nearest neighbors to five, reflecting our prior belief on appropriate neighborhood size. For planning, we set manually the dataset kernel width parameter to $\tau_k^x = 0.04$ and the workflow kernel width parameter to $\tau_k^w = 0.08$ which result on small dataset and workflow neighborhoods respectively. Again, these two parameters were not tuned but simply set on our prior belief of their respective neighborhood size.

(a) Scenario 1, only tested operators.

(b) Scenario 2, tested and untested operators.

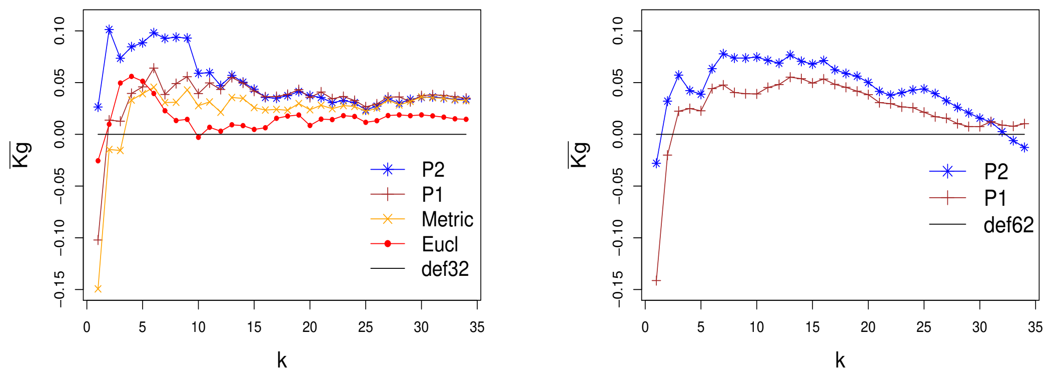

Figure 8: Average correlation gain $\bar{Kg}$ of the different methods against the baseline on the 65 bio-datasets. In the x-axis, $k = 2 \ldots 35$, we have the number of top-k workflows suggested to the user. P1 and P2 are the two planning strategies. Metric and Eucl are baseline methods and def$X$ is the default strategy computed over the set of X workflows.

5.5 Experimental Results

In the following sections we give the results of the experimental evaluation of the different methods we presented so far under the two evaluation scenarios described above.

5.5.1 Scenario 1, Building DM workflows from a Pool of Tested Only Operators

In this scenario, we will evaluate the quality of the DM workflows constructed by the two planning strategies P1 and P2 and compare it to that of the two baseline methods as well as to that of the default strategy. We do leave-one-dataset-out to evaluate the workflow recommendations given by each method. In Figure 8(a) we give the average Kendall gain $\bar{Kg}$ for each method against the default strategy which we compute over their top-k lists for $k = 2, \ldots, 35$. Clearly the P2 strategy is the one that gives the largest improvements with respect to the default strategy, between 5% to 10% of gain, compared to any other method.

We establish the statistical significance of these results for each k, counting the number of datasets for which each method was better/worse than the default, using a McNemar's test. We summarize in Figure 9 the statistical significance results given the p-values for the different ks and give the detailed results in Table 4 in the appendix for all methods for $k = 2 \ldots 35$. We can see that the $P2$ is by far the best method being significantly better than the default for 16 out of the 34 values of k, close to significant ($0.05 < p\text{-value} \leq 0.1$) ten out of the 34 times and never significantly worse. From the other methods, only P2 managed to beat the default and this only for 2 out of the 34 cases of k.

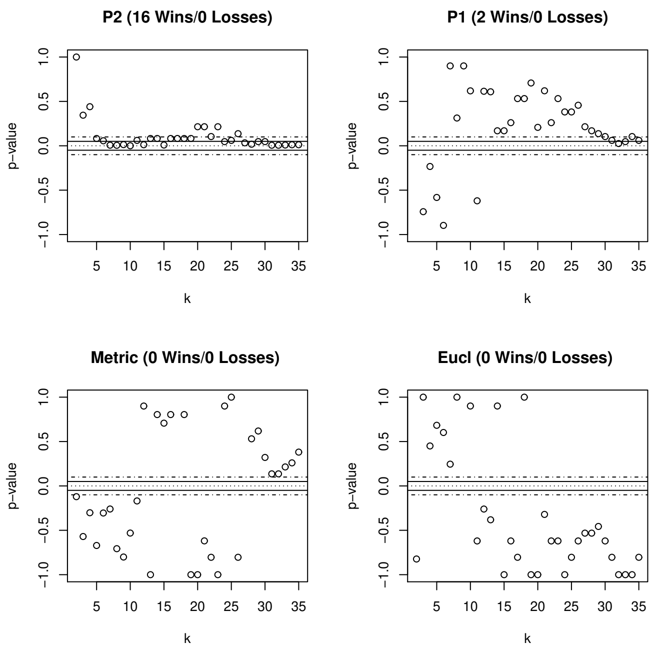

Figure 9: P-values of the McNemar's test on the number of times that the Kendal similarity of a method is better than the default for a given k, Scenario 1. A positive p-value means more wins than losses, a negative the opposite. The solid lines are at $p = +/-0.05$, the dash-dotted at $p = +/-0.1$. The X Wins/ Y Losses in the header indicates the number of times over $k = 3..35$ that the method was significantly better/worse then the default.

When we examine the average accuracy that the top-k workflows suggested by each method achieve, the advantage of P2 is even more striking. Its average accuracy is 1.25% and 1.43% higher than that of the default strategy, for $k = 3$ and $k = 5$ respectively, see Table 3(a). For $k = 3$, P2 achieves a higher average accuracy than the default in 39 out of the 65 datasets, while it under-performs compared to the default only in 20. Using again a McNemar's test the statistical significance is 0.02, i.e. P2 is significantly better than the default strategy when it comes to the average accuracy of the top $k = 3$ workflows it plans; the results are similar for $k = 5$. In fact for the eight top-k lists, $k = 3 \ldots 10$, P2 is significantly better than the default for five values of k, close to significantly better once, and never significantly worse. For the higher values of k, $k = 11 \ldots 35$, it is significantly better 11 times, close to significantly better three times, and never significantly worse. It stops being significantly better when $k > 30$. For such large k values, the average is taken over almost all workflows, thus we do not expect important differences between the lists produced by the different methods. In Figure 10, we visualize the statistical significance results for the different values of $k = 3 \ldots 35$ and give the detailed results in Table 5 of the Appendix.

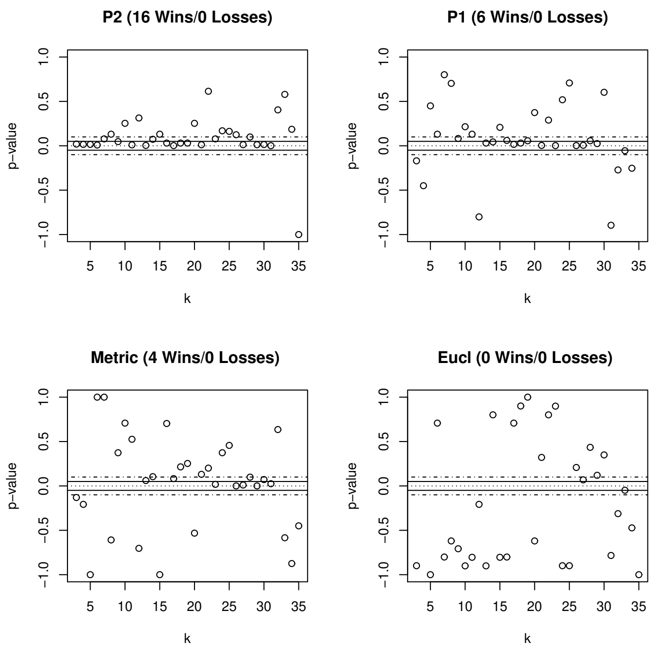

Figure 10: P-values of the McNemar's test on the number of times that the Average Accuracy of a method is better than the default for a given k, Scenario 1. Same figure interpretation as in Figure 9

P1 is never significantly better than the default for all $k = 3 \dots 10$, while for $k = 11 \dots 35$, it is significantly better for nine values of k, close to significantly better three times, and close to significantly worse once. The *Metric* baseline is never significantly better than the default for all $k = 3 \dots 10$, while for $k = 11 \dots 35$, it is significantly better for four values of k,and close to significantly better four times. The results of *EC* are quite poor. In terms of the average accuracy, it is very similar to the default, while in terms of the number of times that it performs better than the default, for most of the cases, it is less than the number of times that it performs worse than the default. As before in Figure 10 we give the results of the statistical significance results for the different values of k and the detailed results in Table 5 of the appendix. P2 is the only method that directly learns and exploits in the workflow planning the associations between the dataset and the workflow characteristics which as the experimental results clearly demonstrate is the strategy that best pays off.

5.5.2 Scenario 2: Building DM workflows from a Pool of Tested and Non-Tested Operators

In the second scenario, we evaluate the performance of the two planning strategies, P1 and P2, where the pool of available operators during the planning is not any more limited to operators with which we have already experimented in the base-level experimented with, but it is extended to include additional operators which are described in the DMOP ontology.

(a) Scenario 1, only tested operators

	$k=3$			$k=5$		
	Avg. Acc	W/L	$p-$value	Avg. Acc	W/L	$p-$value
P2	0.7988	39/20	+0.02	0.7925	41/21	+0.02
P1	0.7886	26/38	0.17	0.7855	35/28	0.45
Metric	0.7861	25/38	0.13	0.7830	32/33	1.00
Eucl	0.7829	30/32	0.90	0.7782	32/33	1.00
def35	0.7863			0.7787		

(b) Scenario 2, tested and untested operators

	$k=3$			$k=5$		
	Avg. Acc	W/L	$p-$value	Avg. Acc	W/L	$p-$value
P2	0.7974	39/24	0.08	0.7907	34/29	0.61
P1	0.7890	29/34	0.61	0.7853	31/34	0.80
def62	0.7867			0.7842		

Table 3: Average accuracy of the top-k workflows suggested by each method. W indicates the number of datasets that a method achieved a top-k average accuracy larger than the respective of the default, and L the number of datasets that it was smaller than the default. $p-$value is the result of the McNemar's statistical significance test; + indicates that the method is statistically better than the default.

We have already described the exact setting in section 5.1; as a reminder the number of possible workflows is now 62. As before, we will estimate performances using leave-one-dataset-out. Note that the two baseline methods, *Metric* and *Eucl*, are not applicable in this setting, since they can be only deployed over workflows with which we have already experimented in the baseline experiments. Here, as we already explained in section 5.3, the default strategy will correspond to the average rank of the 62 possible workflows and we will denote it with *def62*. Note that this is a highly optimistically-biased default method since it relies on the execution of *all* 62 possible workflows on the base-level datasets, unlike P1 and P2 which only get to see 35 workflows for the model building, and the operators therein, but will plan over the larger pool.

In Figure 8(b), we give the average Kendall gain $\bar{Kg}$ for P1 and P2 over the *def62* baseline. Similarly to the first evaluation scenario, P2 has an advantage over P1 since it demonstrates higher gains over the default. Note though that these performance gains are now smaller than they were previously. In terms of the number of k values for which P2 is (close to be) significantly better than the default, these are now six and eight, for the different $k = 2 \dots 35$. *def62* is now once significantly better than P2 and once close to being significantly better. In what concerns P1, there is no significant difference between its performance and *def62*, for any value of k. For values of $k > 30$, P2 systematically under-performs compared to *def62*, due to the advantage of the latter that comes from seeing the performance of all 62 workflows over the base-level dataset. We visualize the statistical significance results in the top row of Figure 11, and give the detailed results in Table 6 of the Appendix for all values of $k = 2 \dots 35$.

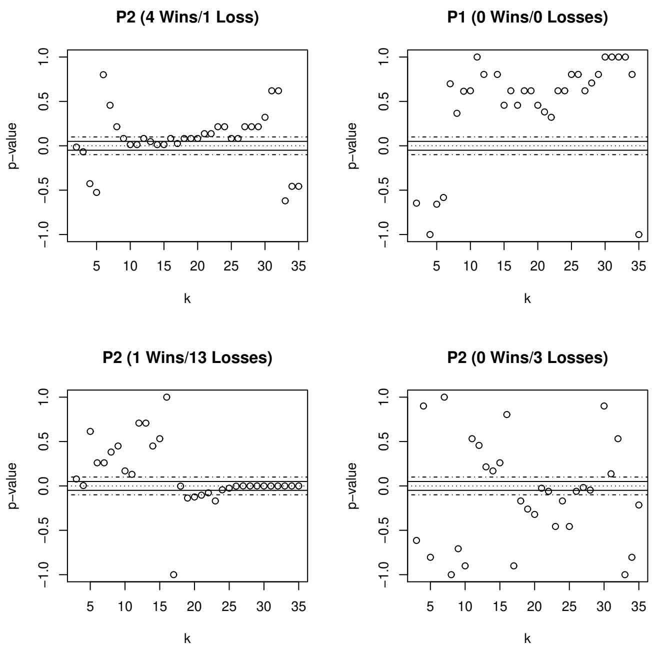

Figure 11: Top row p-values of the McNemar's test on the number of times that the Kendall similarity of a method is better than the default for a given k, Scenario 2. Bottom row same for Average Accuracy. Same figure interpretation as in Figure 9.

In what concerns the performance of P2 with respect to the average accuracy of the top-k workflows it suggests, it has a slight advantage over *def62* but only for very small values of k, up to four. It is significantly better compared to *def62* only once, $k = 4$. For $k = 5$ to 17, the two methods have no significant difference, while for $k = 18 \dots 35$ P2 is worse, 13 times in a significant manner. For P1 the picture is slightly different, its average accuracy is not significantly different than *def62*, with the exception of three k values for which it is significantly worse. We visualize the statistical significance results in the bottom row of Figure 11, and give the detailed results in Table 7. It seems that the fact that P2 learns directly the associations between datasets and workflow characteristics puts it at a disadvantage when we want to plan over operators that have not been tested in the training phase. In such a scenario, the P1 strategy which weights preferences by dataset similarity and by workflow similarity seems to cope better with untested operators. Nevertheless it is still not possible to outperform the default strategy in a significant manner, keeping however in mind that *def62* is an optimistic default strategy because it is based on the experimentation of all possible workflows on the training dataset.

5.6 Discussion

In the previous sections, we evaluated the two workflow planning strategies in two settings: planning only over tested operators, and planning with both tested and untested operators.

In the first scenario, the P2 planning strategy that makes use of the heterogeneous metric learning model, which directly connects dataset to workflow characteristics, clearly stands out. It outperforms the default strategy in terms of the Kendall Similarity gain, in a statistically significant, or close to statistically significant, manner for 24 values of $k \in [2, \ldots, 35]$; in terms of the average accuracy of its top-k workflows, it outperforms it for 20 values of k in a statistically significant, or close to statistically significant, manner. All the other methods, including P1, follow with a large performance difference from P2.

When we allow the planners to include in the workflows operators which have not been used in the baseline experiments, but which are annotated in the DMOP ontology, P2's performance advantage is smaller. In terms of the Kendall similarity gain, this is statistically significant, or close to, for $k \in [10, \ldots, 20]$. With respect to the average accuracy of its top-k lists, this is better than the default only for very small lists, $k = 3, 4$; for $k > 23$, it is in fact significantly worse. P1 fairs better in the second scenario, however its performance is not different from the default method. Keep in mind though that the baseline used in the second scenario is a quite optimistic one.

In fact, what we see is that we are able to generalize and plan well over datasets, as evidenced by the good performance of P2 in the first setting. However, when it comes to generalizing both over datasets and operators as it is the case for the second scenario the performance of the planned workflows is not good, with the exception of the few top workflows. If we take a look at the new operators we added in the second scenario, these were a feature selection algorithm, Information Gain Ratio, and four classification algorithms, namely a Linear Discriminant Algorithm, the Ripper rule induction algorithm, a Neural Net algorithm, and a Random Tree. Out of them, only for Information Gain Ratio we have seen during the base level set of experiments an algorithm, Information Gain, that has a rather similar learning bias to it. The Ripper rule induction is a sequential covering algorithm, the closest operators to which in our set of training operators are the two decision tree algorithms which are recursive partitioning algorithms. With respect to the DMOP ontology, *Ripper* shares a certain number of common characteristics with decision trees, however the meta-mining model contains no information on how the set-covering learning bias performs over different datasets. This might lead to it being selected for a given dataset based on its common features with the decision trees, while its learning bias is not in fact appropriate for that dataset. Similar observations hold for the other algorithms, for example *LDA* shares a number of properties with SVM_l and *LR*, however its learning bias, maximizing the between-to-within class distances ratio, is different from the learning biases of these two, as before the meta-mining model contains no information on how its bias associates with dataset characteristics.

Overall, the extent to which the system will be able to plan, over tested and untested operators, workflows that achieve a good performance, depends on the extend to which the properties of the latter have been seen during the training of the meta-mining models within the operators with which we have experimented with, as well as on whether the unseen properties affect critically the final performance. In the case that all operators are well characterized experimentally, as we did in Scenario 1, then the performance of the workflows designed by the P2 strategy is very good. Note that it is not necessary that all operators or workflows are applied to all datasets, it is enough to have a sufficient set of experiments for each operator. The heterogeneous metric learning algorithm can handle

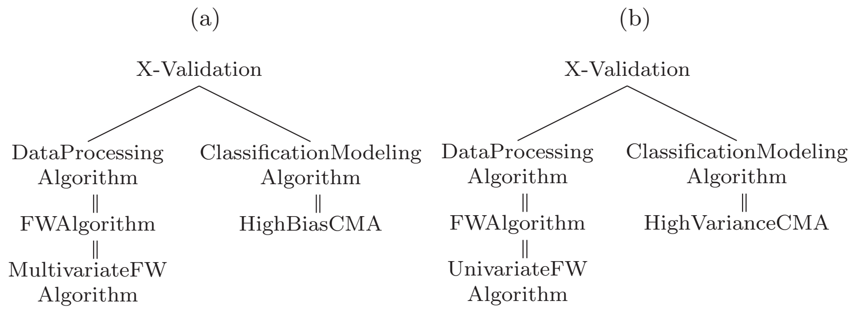

Figure 12: Top-ranked workflow patterns according to their average absolute weights given in matrix $\mathbf{V}$.

incomplete preference matrices, using only the available information. Of course it is clear that the more the available information, whether in the form of complete preference matrices or in the form of extensive base-level experiments over large number of datasets, the better the quality of the learned meta-mining model will be. It will be interesting to explore the sensitivity of the heterogeneous metric learning method over different levels of completeness of the preference matrix; however this is outside the scope of the present paper.

We can quantify the importance of the different workflow patterns and that of the operators' properties by analyzing the linear transformation over the workflow patterns contained in the heterogeneous metric. More precisely, we establish the learned importance of each workflow pattern by averaging the absolute values of the weights it is assigned over the different factors (rows) of the $\mathbf{V}$ linear transformation matrix of Eq.(11). Note that under this approach, we only establish the importance of the patterns, and not whether they are associated with good or bad predictive performance. In Figure 12, we give the two most important patterns as these are determined on the basis of their averaged absolute weights. Both of them describe relations between the workflow operators, the first one indicates that we have a multivariate feature weighting algorithm followed by a high bias classification algorithm, while the second describes a univariate feature weighting algorithm followed by a high bias classification algorithm. A systematic analysis of the learned model could provide hints on where one should focus the ontology building effort, looking at what are the important patterns as well as what are the patterns that are not used. In addition, it can reveal which parts of the ontology might need refinement in order to distinguish between different workflows with respect to their expected performance.

6. Conclusions and Future Work

In this paper, we have presented what is, to the best of our knowledge, the first system that is able to plan data mining workflows, for a given task and a given input dataset, that are expected to optimize a given performance measure. The system relies on the tight interaction of a hierarchical task network planner with a learned meta-mining model to plan the workflows. The meta-mining model, a heterogeneous learned metric, associates datasets characteristics with workflow characteristics which are expected to lead to good performance. The workflow characteristics describe relations between the different components of the workflows, capturing global interactions of the various operators that appear within them, and incorporate domain knowledge as the latter is given in a data mining ontology (DMOP). We learn the meta-mining model on a collection of past base-level mining experiments, data mining workflows applied on different datasets. We carefully evaluated the system on the task of classification and we showed that it outperforms in a significant manner a number of baselines and the default strategy when it has to plan over operators with which we have experimented with in the base-level experiments. The performance advantage is less pronounced when it has to consider also during planning operators with which we have not experimented with in the base-level experiments, especially when the properties of these operators were not present within other operators with which we have experimented with in the base-level experiments.

The system is directly applicable to other mining tasks e.g. regression, clustering. The reasons for which we focused on classification were mainly practical: there is extensive annotation of the classification task and the related concepts in the data mining ontology, large availability of classification datasets, and extensive relevant work on meta-learning and dataset characterization for classification. The main hurdle in experimenting with a different mining task is the annotation of the necessary operators in the DMOP ontology and the set up of a base-level collection of mining experiments for the specific task. Although the annotation of new algorithms and operators is a quite labor intensive task, many of the concepts currently available in the DMOP are directly usable in other mining tasks, e.g. cost functions, optimization problems, feature properties etc. In addition, there is a small active community, the DMO-foundry[6], maintaining and augmenting collaboratively the ontology with new tasks and operators, significantly reducing the deployment barrier for a new task. In the DMO-foundry web site, one can find a number of tools and templates to facilitate the addition of new concepts and operators in the ontology as well as to annotate the existing ones. Having said that we should note that the use of DMOP is not sine-qua-non for the system to function. We can very well perform the workflow characterization task by mining just over ground operators, without using the ontology. The downside of that would be that the extracted patterns will not be generalized nor will they contain operator properties. Instead they will be defined over ground operators. Everything else remains as it is.

There are a number of issues that we still need to explore in a finer detail. We would like to gain a deeper understanding and a better characterization of the reduced performance in planning over untested operators; for example, under what conditions we can be relatively confident on the suitability of an untested operator within a workflow. We want to experiment with the strategy we suggested for parameter tuning, in which we treat the parameters as yet another property of the operators, in order to see whether it gives better results; we expect it will. We want to study in detail how the level of missing information in the preference matrix affects the performance of the system, as well as whether using ranking based loss functions in the metric learning problem instead of sum of squares would lead to even better performance.

6. http://www.dmo-foundry.org/

On a more ambitious level we want to bring in ideas from reinforcement learning (Sutton & Barto, 1998); let the system design its own workflows in a systematic way and have them applied on the collection of available datasets in order to derive even better characterizations of the workflow space and how they relate to the dataset space, exploring for example areas in which the meta-mining model is less confident.

Acknowledgments

This work has been partially supported by the European Community 7^{th} framework program ICT-2007.4.4 under grant number 231519 "e- Lico: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science". Alexandros Kalousis was partially supported by the RSCO ISNET NFT project. The basic HTN planner has been the result of collaborative work within the e-LICO project of Jörg-Uwe Kietz, Floarea Serban, Simon Fischer. We would like to thank Jun Wang for his important contribution in developing the metric learning part of the paper. In addition we would like to thank all the members of the AI lab, Adam Woznica, Huyen Do, and Jun Wang, for the significant effort they placed in providing content in the DMOP. Finally, we would like to thank the reviewers for the suggestions that helped improve the paper.

Appendix A. Detailed Results

	P2			P1			Metric			Eucl		
k	W	L	p-value	W	L	p-value	W	L	p-value	W	L	p-value
2	8	7	1	13	13	1	4	11	0.121	9	11	0.823
3	17	11	0.344	17	20	0.742	12	16	0.570	16	15	1.000
4	24	18	0.440	18	27	0.233	19	27	0.302	25	19	0.450
5	35	21	0.082	24	29	0.582	23	27	0.671	29	25	0.683
6	39	23	0.056	29	31	0.897	26	35	0.305	32	27	0.602
7	41	19	**0.006**	33	31	0.900	27	37	0.260	35	25	0.245
8	43	20	**0.005**	36	27	0.313	30	34	0.707	31	30	1.000
9	43	22	**0.013**	33	31	0.900	30	33	0.801	32	32	1.000
10	47	18	**0.000**	35	30	0.619	29	35	0.531	33	31	0.900
11	40	24	0.060	30	35	0.619	26	38	0.169	30	35	0.619
12	42	21	0.011	34	29	0.614	33	31	0.900	27	37	0.260
13	40	25	0.082	33	28	0.608	32	33	1.000	28	36	0.381
14	40	25	0.082	38	26	0.169	34	31	0.804	33	31	0.900
15	43	21	**0.008**	38	26	0.169	34	30	0.707	32	33	1.000
16	40	25	0.082	37	27	0.260	34	31	0.804	30	35	0.619
17	40	25	0.082	35	29	0.531	32	32	1.000	31	34	0.804
18	40	25	0.082	35	29	0.531	34	31	0.804	33	32	1.000
19	40	25	0.082	34	30	0.707	32	33	1.000	32	33	1.000
20	38	27	0.214	37	26	0.207	31	32	1.000	32	33	1.000
21	38	27	0.214	35	30	0.619	30	35	0.619	28	37	0.321
22	39	25	0.104	37	27	0.260	31	34	0.804	30	35	0.619
23	38	27	0.214	35	29	0.531	32	33	1.000	30	35	0.619
24	41	24	**0.047**	36	28	0.381	33	31	0.900	32	33	1.000
25	40	24	0.060	36	28	0.381	33	32	1.000	31	34	0.804
26	39	26	0.136	36	29	0.456	31	34	0.804	30	35	0.619
27	41	23	**0.033**	38	27	0.214	32	32	1.000	29	35	0.531
28	42	22	**0.017**	38	26	0.169	35	29	0.531	29	35	0.531
29	41	24	**0.047**	39	26	0.136	35	30	0.619	29	36	0.456
30	41	24	**0.047**	39	25	0.104	37	28	0.321	30	35	0.619
31	44	21	**0.006**	40	24	0.060	39	26	0.136	31	34	0.804
32	44	21	**0.006**	42	23	**0.025**	39	26	0.136	32	33	1.000
33	43	21	**0.008**	41	24	**0.047**	38	27	0.214	32	33	1.000
34	42	21	**0.011**	39	25	0.104	37	27	0.260	32	33	1.000
35	42	21	**0.011**	40	24	0.060	36	28	0.381	31	34	0.804
Total	Wins 16/Losses 0			Wins 2/Losses 0			Wins 0/Losses 0			Wins 0/Losses 0		

Table 4: Wins/Losses and respective P-values of the McNemar's test on the number of times that the Kendal similarity of a method is better than the Kendal similarity of the default, Scenario 1. In bold, the winning p-value that are lower than 0.05.

	P2				P1				Metric				EC				Def
k	Avg.Acc	W	L	p-value	Avg.Acc	W	L	p-value	Avg.Acc	W	L	p-value	Avg.Acc	W	L	p-value	Avg.Acc
3	0.798	39	20	**0.019**	0.788	26	38	0.169	0.786	25	38	0.130	0.782	30	32	0.898	0.786
4	0.793	41	21	**0.015**	0.786	28	35	0.449	0.784	26	37	0.207	0.780	32	32	1	0.785
5	0.792	41	21	**0.015**	0.785	35	28	0.449	0.783	32	33	1	0.778	32	33	1	0.778
6	0.789	43	21	**0.008**	0.785	38	25	0.130	0.782	33	32	1	0.777	34	30	0.707	0.777
7	0.786	39	24	0.077	0.786	33	30	0.801	0.780	32	31	1	0.776	30	33	0.801	0.780
8	0.784	38	25	0.130	0.783	33	29	0.703	0.779	28	33	0.608	0.774	30	35	0.619	0.778
9	0.782	41	24	0.047	0.782	40	25	0.082	0.779	35	27	0.374	0.773	30	34	0.707	0.775
10	0.780	36	26	0.253	0.781	38	27	0.214	0.778	34	30	0.707	0.773	31	33	0.900	0.775
11	0.778	41	20	**0.010**	0.778	38	25	0.130	0.776	34	28	0.525	0.773	31	34	0.804	0.774
12	0.777	36	27	0.313	0.777	30	33	0.801	0.775	29	33	0.703	0.772	26	37	0.207	0.774
13	0.777	44	20	**0.004**	0.777	40	22	**0.030**	0.774	40	24	0.060	0.772	31	33	0.900	0.771
14	0.774	38	23	0.073	0.775	40	23	0.043	0.773	39	25	0.104	0.772	33	30	0.801	0.771
15	0.773	38	25	0.130	0.774	37	26	0.207	0.771	32	33	1	0.771	31	34	0.804	0.771
16	0.772	40	22	**0.030**	0.772	40	24	0.060	0.770	33	29	0.703	0.770	30	33	0.801	0.769
17	0.771	43	18	**0.002**	0.771	41	21	**0.015**	0.770	40	25	0.082	0.769	34	30	0.707	0.766
18	0.770	40	22	**0.030**	0.770	40	22	**0.030**	0.769	38	27	0.214	0.767	33	31	0.900	0.765
19	0.769	40	22	**0.030**	0.769	39	23	0.056	0.767	36	26	0.253	0.767	32	31	1	0.765
20	0.767	36	26	0.253	0.768	35	27	0.374	0.767	29	35	0.531	0.766	30	35	0.619	0.766
21	0.766	42	21	**0.011**	0.767	41	18	**0.004**	0.766	38	25	0.130	0.765	37	28	0.321	0.763
22	0.765	34	29	0.614	0.765	33	24	0.289	0.765	36	25	0.200	0.764	33	30	0.801	0.763
23	0.763	39	24	0.077	0.763	45	19	0.001	0.764	42	22	0.017	0.762	32	30	0.898	0.761
24	0.762	38	26	0.169	0.762	33	27	0.518	0.763	35	27	0.374	0.761	30	32	0.898	0.761
25	0.761	37	25	0.162	0.761	34	30	0.707	0.762	36	29	0.456	0.760	30	32	0.898	0.760
26	0.760	37	24	0.124	0.760	43	18	0.002	0.761	45	11	**0.001**	0.758	37	26	0.207	0.756
27	0.759	39	19	**0.012**	0.758	43	20	**0.005**	0.759	41	20	**0.010**	0.757	37	22	0.068	0.756
28	0.757	33	20	0.099	0.757	39	23	0.056	0.758	38	24	0.098	0.756	33	26	0.434	0.756
29	0.755	32	14	**0.012**	0.754	34	17	**0.025**	0.755	35	10	**0.000**	0.754	31	19	0.119	0.753
30	0.753	33	15	**0.014**	0.751	32	27	0.602	0.752	33	19	0.071	0.751	32	24	0.349	0.751
31	0.751	32	10	**0.001**	0.748	28	30	0.895	0.750	34	17	**0.025**	0.749	25	28	0.783	0.749
32	0.749	21	15	0.404	0.746	22	31	0.271	0.748	22	18	0.635	0.747	20	28	0.312	0.748
33	0.747	8	5	0.579	0.744	16	30	0.055	0.745	13	17	0.583	0.745	12	25	0.048	0.747
34	0.742	18	10	0.185	0.741	26	36	0.253	0.742	19	21	0.874	0.742	13	18	0.472	0.742
35	0.737	2	3	1	0.737	2	2	1	0.737	2	5	0.449	0.737	2	3	1	0.737
Total	Wins 16/Losses 0				Wins 6/0 Losses				Wins 4/Losses 0				Wins 0/ Losses 0				

Table 5: Average Accuracy, Wins/Losses, and respective P-values of the McNemar's test on the number of times the Average Accuracy of a method is better than the Average Accuracy of the default, Scenario 1. In bold, the winning *p*-value that are lower than 0.05.

	P2			P1		
k	W	L	p-value	W	L	p-value
2	9	24	0.014	8	11	0.646
3	15	28	0.067	13	13	1.000
4	25	32	0.426	17	18	1.000
5	28	34	0.525	21	25	0.658
6	33	30	0.801	24	29	0.582
7	36	29	0.456	32	28	0.698
8	38	27	0.214	34	26	0.366
9	40	25	0.082	34	29	0.614
10	43	22	0.013	35	30	0.619
11	43	22	0.013	33	32	1.000
12	40	25	0.082	34	31	0.804
13	41	24	**0.047**	32	32	1.000
14	43	22	**0.013**	34	31	0.804
15	43	22	**0.013**	36	29	0.456
16	40	25	0.082	35	30	0.619
17	42	23	**0.025**	36	29	0.456
18	40	25	0.082	35	30	0.619
19	40	25	0.082	35	30	0.619
20	40	25	0.082	36	29	0.456
21	39	26	0.136	36	28	0.381
22	39	26	0.136	37	28	0.321
23	38	27	0.214	35	30	0.619
24	38	27	0.214	35	30	0.619
25	40	25	0.082	34	31	0.804
26	40	25	0.082	34	31	0.804
27	38	27	0.214	35	30	0.619
28	38	27	0.214	34	30	0.707
29	38	27	0.214	34	31	0.804
30	37	28	0.321	33	32	1.000
31	35	30	0.619	33	32	1.000
32	35	30	0.619	33	32	1.000
33	30	35	0.619	33	32	1.000
34	29	36	0.456	34	31	0.804
35	29	36	0.456	32	33	1.000
Total	Wins 4/Losses 0			Wins 0/Losses 0		

Table 6: Wins/Losses and P-values of the McNemar's test on the number of times Kendal similarity of a method is better than the Kendal similarity of the default, Scenario 2. In bold, the winning p-value that are lower than 0.05.

	P2				P1				Def
k	Avg.Acc	W	L	p-value	Avg.Acc	W	L	p-value	Avg.Acc
3	0.797	39	24	0.077	0.789	29	34	0.614	0.786
4	0.792	44	20	**0.004**	0.784	33	31	0.900	0.780
5	0.790	34	29	0.614	0.785	31	34	0.804	0.784
6	0.787	37	27	0.260	0.782	32	32	1.000	0.778
7	0.785	37	27	0.260	0.783	33	32	1.000	0.778
8	0.783	36	28	0.381	0.783	32	33	1.000	0.779
9	0.782	35	28	0.449	0.783	30	34	0.707	0.778
10	0.781	38	26	0.169	0.784	31	33	0.900	0.776
11	0.779	38	25	0.130	0.782	35	29	0.531	0.773
12	0.777	34	30	0.707	0.780	36	29	0.456	0.772
13	0.775	34	30	0.707	0.779	38	27	0.214	0.770
14	0.774	35	28	0.449	0.779	38	26	0.169	0.770
15	0.773	35	29	0.531	0.777	37	27	0.260	0.770
16	0.773	33	32	1.000	0.776	34	31	0.804	0.770
17	0.772	32	33	1.000	0.774	31	33	0.900	0.771
18	0.770	19	44	0.002	0.773	26	38	0.169	0.773
19	0.769	26	39	0.136	0.772	27	37	0.260	0.772
20	0.768	24	37	0.124	0.772	28	37	0.321	0.771
21	0.768	25	39	0.104	0.770	23	42	0.025	0.770
22	0.767	24	39	0.077	0.770	24	40	0.060	0.771
23	0.767	26	38	0.169	0.769	29	36	0.456	0.769
24	0.766	23	40	0.043	0.768	26	38	0.169	0.768
25	0.765	23	42	0.025	0.768	29	36	0.456	0.767
26	0.763	20	45	0.002	0.767	24	40	0.060	0.768
27	0.762	14	51	0.000	0.768	22	42	0.017	0.769
28	0.762	15	50	0.000	0.767	24	41	0.047	0.768
29	0.761	16	48	0.000	0.767	32	32	1.000	0.766
30	0.760	18	47	0.000	0.766	33	31	0.900	0.764
31	0.759	18	47	0.000	0.766	39	26	0.136	0.763
32	0.757	18	47	0.000	0.765	35	29	0.531	0.764
33	0.756	17	48	0.000	0.765	33	33	1.000	0.764
34	0.756	16	49	0.000	0.764	31	34	0.804	0.764
35	0.755	13	51	0.000	0.763	27	38	0.214	0.764
Total	Wins 1/Losses 13				Wins 0/Losses 3				

Table 7: Avg.Acc., Wins/Losses, and respective P-values of the McNemar's test on the number of times the Average Accuracy of a method is better than the Average Accuracy of the default, Scenario 2. In bold, the winning *p*-value that are lower than 0.05.

References

Bernstein, A., Provost, F., & Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, *17*(4), 503–518.

Bose, R. J. C., & der Aalst, W. M. V. (2009). Abstractions in process mining: A taxonomy of patterns. In *Proceedings of the 7th International Conference on Bussiness Process Management*.

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2008). *Metalearning: Applications to Data Mining* (1 edition). Springer Publishing Company, Incorporated.

Bringmann, B. (2004). Matching in frequent tree discovery. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM04*, pp. 335–338.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide. Tech. rep., The CRISP-DM consortium.

Fagin, R., Kumar, R., & Sivakumar, D. (2003). Comparing top k lists. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pp. 28–36, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37.

Forbes, J., & Andre, D. (2000). Practical reinforcement learning in continuous domains. Tech. rep., Berkeley, CA, USA.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., & Myers, J. (2007). Examining the challenges of scientific workflows. *Computer*, *40*(12), 24–32.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, *11*(1), 10–18.

Hilario, M. (2002). Model complexity and algorithm selection in classification. In *Proceedings of the 5th International Conference on Discovery Science*, DS '02, pp. 113–126, London, UK, UK. Springer-Verlag.

Hilario, M., & 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, pp. 180–191, London, UK, UK. Springer-Verlag.

Hilario, M., Kalousis, A., Nguyen, P., & Woznica, A. (2009). A data mining ontology for algorithm selection and meta-learning. In *Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*.

Hilario, M., Nguyen, P., Do, H., Woznica, A., & Kalousis, A. (2011). Ontology-based meta-mining of knowledge discovery workflows. In Jankowski, N., Duch, W., & Grabczewski, K. (Eds.), *Meta-Learning in Computational Intelligence*. Springer.

Ho, T. K., & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*(3), 289–300.

Ho, T. K., & Basu, M. (2006). *Data complexity in pattern recognition*. Springer.

Hoffmann, J. (2001). Ff: The fast-forward planning system. *AI magazine*, *22*(3), 57.

Kalousis, A. (2002). *Algorithm Selection via Metalearning*. Ph.D. thesis, University of Geneva.

Kalousis, A., Gama, J., & Hilario, M. (2004). On data and algorithms: Understanding inductive performance. *Machine Learning*, *54*(3), 275–312.

Kalousis, A., & Theoharis, T. (1999). Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intell. Data Anal.*, *3*(5), 319–337.

Kietz, J.-U., Serban, F., Bernstein, A., & Fischer, S. (2009). Towards Cooperative Planning of Data Mining Workflows. In *Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-09)*.

Kietz, J.-U., Serban, F., Bernstein, A., & Fischer, S. (2012). Designing kdd-workflows via htn-planning for intelligent discovery assistance. In *5th PLANNING TO LEARN WORKSHOP WS28 AT ECAI 2012*, p. 10.

King, R. D., Feng, C., & Sutherland, A. (1995). Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, *9*(3), 289–333.

Klinkenberg, R., Mierswa, I., & Fischer, S. (2007). Free data mining software: Rapidminer 4.0 (formerly yale). `http://www.rapid-i.com/`.

Köpf, C., Taylor, C., & Keller, J. (2000). Meta-analysis: From data characterisation for meta-learning to meta-regression. In *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support,Meta-Learning and ILP*.

Kramer, S., Lavrač, N., & Flach, P. (2000). Relational data mining.. chap. Propositionalization Approaches to Relational Data Mining, pp. 262–286. Springer-Verlag New York, Inc., New York, NY, USA.

Leite, R., & Brazdil, P. (2010). Active testing strategy to predict the best classification algorithm via sampling and metalearning. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pp. 309–314, Amsterdam, The Netherlands, The Netherlands. IOS Press.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). Pddl-the planning domain definition language..

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (1994). Machine learning, neural and statistical classification..

Nguyen, P., Kalousis, A., & Hilario, M. (2011). A meta-mining infrastructure to support kd workflow optimization. *Proc of the ECML/PKDD11 Workshop on Planning to Learn and Service-Oriented Knowledge Discovery*, 1.

Nguyen, P., Kalousis, A., & Hilario, M. (2012a). Experimental evaluation of the e-lico meta-miner. In *5th PLANNING TO LEARN WORKSHOP WS28 AT ECAI 2012*, p. 18.

Nguyen, P., Wang, J., Hilario, M., & Kalousis, A. (2012b). Learning heterogeneous similarity measures for hybrid-recommendations in meta-mining. In *IEEE 12th International Conference on Data Mining (ICDM)*, pp. 1026 –1031.

Peng, Y., Flach, P. A., Soares, C., & Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In *Discovery Science*, pp. 141–152. Springer.

Pfahringer, B., Bensusan, H., & Giraud-Carrier., C. (2000). Meta-learning by landmarking various learning algorithms.. *Proc. 17th International Conference on Machine Learning*, 743–750.

R Core Team (2013). R: A language and environment for statistical computing. `http://www.R-project.org/`.

Smart, W. D., & Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 903–910, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Soares, C., & Brazdil, P. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '00, pp. 126–135, London, UK. Springer-Verlag.

Srebro, N., Rennie, J. D. M., & Jaakkola, T. S. (2005). Maximum-margin matrix factorization. In Saul, L. K., Weiss, Y., & Bottou, L. (Eds.), *Advances in Neural Information Processing Systems 17*, pp. 1329–1336. MIT Press, Cambridge, MA.

Sutton, R., & Barto, A. (1998). Reinforcement learning: An introduction. *Neural Networks, IEEE Transactions on*, *9*(5), 1054.

Van der Aalst, W. M., & Giinther, C. (2007). Finding structure in unstructured processes: The case for process mining. In *Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on*, pp. 3–12. IEEE.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, *5*(04), 597–604.

Zaki, M. J. (2005). Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, *17*(8), 1021–1035. special issue on Mining Biological Data.

Záková, M., Kremen, P., Zelezny, F., & Lavrac, N. (2011). Automating knowledge discovery workflow composition through ontology-based planning. *Automation Science and Engineering, IEEE Transactions on*, *8*(2), 253–264.