FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DataMining4Dummies: A web application for automatic selection of data mining algorithms for new problems

Tiago Miguel Moreira Pereira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares (PhD)

Second Supervisor: Pavel Bernard Brazdil (PhD)

March 3, 2014

DataMining4Dummies: A web application for automatic selection of data mining algorithms for new problems

Tiago Miguel Moreira Pereira

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Rui Carlos Camacho de Sousa Ferreira da Silva (PhD)
External Examiner: José Manuel Matos Moreira (PhD)
Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares (PhD)
Co-Supervisor: Pavel Bernard Brazdil (PhD)

March 3, 2014

Abstract

The interest in the area of classification and prediction is growing rapidly in industry and commerce. A large number of data mining tools are already available. However, such tools are still of limited use to end-users who are not experts. This is due to the fact that machine learning algorithms are non-trivial. As a result, users of machine learning/data mining systems are faced with two major problems: selecting the most suitable algorithm to use on a given dataset, and combining this algorithm with useful and effective transformations of the data. Traditionally, these problems are solved by trial-and-error or consulting experts. The first solution is time consuming and unreliable, while the second is expensive and based on preferences of the experts. Clearly, neither solution is completely satisfactory for the non-expert end-users. Therefore automatic and systematic guidance is required.

By analysing the state of the art we can see how different attempts have been made to address this problem. Some examples are StatLog, ML Toolbox, METAL and DM Advisor, to name a few. Although some of them have shown very interesting results, they are still tool restricted and present a lack of satisfactory user guidance, simplicity and process transparency. In DM Advisor, two of the main shortcomings are the systems scalability and the project did not took full advantage of its distributed approach.

The focus of this dissertation is to improve support to machine learning/data mining end-users, by creating a new system that will allow the recommendation and use of the most promising algorithms in a distributed and collaborative way. In this thesis we rebuild DM Advisor and integrated with OpenML, by collecting its most significant meta-features and applying a meta-learning technique to learn from the collected experiments. The distributed concept introduced by DM Advisor was also improved to take fully advantage of the distributed execution strategy. Finally, the developed system also has the ability of sharing data with others through OpenML.

Resumo

O interesse na área de classificação e previsão está a crescer rapidamente na indústria e no comércio e uma série de ferramentas de data mining já estão disponíveis. No entanto essas ferramentas ainda são de utilidade limitada para os utilizadores finais que não sejam especialistas. Isto é devido ao facto de os algoritmos de aprendizagem não serem triviais. Como resultado, utilizadores de machine learning/data mining são confrontados com dois desafios: escolher qual o algoritmo mais adequado para usar num determinado conjunto de dados, e combiná-los com transformações úteis e eficazes aos dados. Tradicionalmente, este tipo de problemas é resolvido através de tentativa-e-erro ou consultando especialistas. A primeira solução é demorada e pouco fiável; enquanto que a segunda é dispendiosa e depende das preferências do perito. Claramente, nenhuma das soluções é completamente satisfatória para utilizadores finais não-especialistas. Portanto, é necessária uma orientação automática e sistemática é necessária.

Ao analisar o estado da arte, podemos ver como foram desenvolvidas diversas tentativas para abordar este problema, nomeadamente Statlog, ML Toolbox, METAL e DM Advisor, para citar alguns. Apesar de algumas dessas tentativas já demonstrarem resultados muito interessantes, as mesmas ainda são dependentes de ferramentas específicas e apresentam alguma falta de orientação, simplicidade e transparência no processo. No DM Advisor, duas das principais carências são a sua escalabilidade e o project não aproveitou completamente a sua abordagem distribuída.

O foco desta dissertação é trazer uma nova abordagem para os utilizadores, através da criação de um novo sistema que permitirá a recomendação e o uso dos modelos/algoritmos mais promissores de uma forma distribuída e colaborativa. Esta tese reconstruiu o DM Advisor e o integrou com o OpenML, recolhendo as meta-características significativas e aplicar uma técnica de meta-learning para aprender com as experiências recolhidas. Também foi melhorado o conceito distribuído introduzido pelo DM Advisor para aproveitar plenamente a estratégia de execução distribuída. Finalmente, o sistema desenvolvido também tem a capacidade de partilhar dados através do OpenML.

Acknowledgements

No matter what, were or who, in every moment of our lives, people will always leave their marks on us, and the longer the journey the bigger will be number of marks. Being at the end of one of the most important journeys I will ever do in my life, the conclusion of this dissertation, it is not the *finish line* that will make it memorable, but the travel itself and the people who did it with me.

Firstly, I would like to start in expressing my gratitude to my supervisors. Professor Carlos Soares, for his effort in being always available, and the patience with all my questions. Professor Pavel Brazdil, for his knowledge, interest and experience he brought to this thesis.

I would also like to thank Professor Jaime Villate, one of the best professors I have ever met, who influenced me in the most positive way possible and helped me more than I can thank.

The words "thank you" sound so little when to express my gratitude to my family. My mother for her constant joy in seeing her son fulfil his dream. My father who supported me through all these years just to see his son happy. My brother, that no matter what, was always there to give me his strength and his smile, to motivate me the best he could.

To all my friends who accompanied me during this travel, either by sharing the same sleepless nights, or simply just being there.

And finally, to my girlfriend. Who always believed in me, listened to me and would always, somehow, put a smile on my face. She is one of most inspiring persons I have ever met and who I will always treasure with me, no matter what the roads lies ahead.

My gratitude to you all, and may this journey prove itself worthy...

Project "NORTE-07-0124-FEDER-000059" is funded by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

List of Figures

Abbreviations

ADIDA	Automated Distributed Intelligent Discovery Assistant
API	Application Programming Interface
ARFF	Attribute-Relation File Format
CRUD	Create, Read, Update and Delete
CSV	Comma-Separated Values
DAS	Default Algorithm Server
DSM	Domain-Specific Modelling
DAS	Default Algorithm Server
DM	Data Mining
DMA	Data Mining Advisor
DM4D	Data Mining for Dummies
EDA	Exploratory Data Analysis
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBL	Instance-Based Learning
IDA	Intelligent Discovery Assistant
IP	Internet Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KDD	Knowledge-Discovery in Databases
k-NN	k-Nearest Neighbours
MDE	Model-Driven Engineering
MVC	Model-View-Controller
ML	Machine Learning
OS	Operating System
REST	Representational State Transfer
RMSE	Root Mean Squared Error
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

> "Living in the 'Information Age', we are facing a deluge of digital data being generated at ever higher speeds. However, to make this data useful for decision makers, one first needs to make sense of it" [SVB12, chap. Introduction].

Research and industry increasingly make use of large amounts of data to support decision-making. However, data typically needs to be analysed in a non-trivial refinement process, which requires technical expertise about the methods and algorithms [SVB12].

Novice data miners are usually overwhelmed by the plethora of algorithms available on current data-mining (DM) and machine learning (ML) tools. Given a problem, they have no idea which methods can be used because such tools lack guidelines to select the right method for the problem under analysis. This is because the formulation of precise guidelines is often difficult or even impossible [Sch94, Wol01] and specialists rely on years of accumulated experience, which is hard to express explicitly [NT95]. Aggravating the problem is the ever-increasing number of operators [Han97] that few experts can keep up with. Obviously, "*(...) there is a dire need to support both experts and novices in their data analysis tasks*" [SVB12].

Furthermore real world applications are generally time-sensitive. Thus, given the lack of time practitioners and researchers tend to use only a few algorithms. By analysing the entries from the KDD-cup (a data analysis competition) we can see that the majority of experts use no more than two methods: a decision tree learner; and their own induction method [KBF+00], hoping that the set of assumptions embedded in these algorithms will match the characteristics of the data [BGCSV08].

Traditionally, this kind of support is provided by experts/consultants. These are, however, often not available and also mired by the growth of methods. To address this issue, a number of researchers have proposed the use of automated systems (e.g., [Eng96], [BPH05], [CDCS08]). The general idea is to build a system that advises users in all stages of a data analysis process [FPSS96b] — essentially, the provision of an automated Intelligent Discovery Assistant (IDA) [BPH05].

A potential source of support for algorithm recommendation are repositories of experimental data, such as MyExperiment[1] or OpenML[2]. OpenML is a particularly interesting repository because it contains an extensive amount of data describing learning problems and the performance of each algorithms on those problems. OpenML also provides some tools to manually explore the data and these data probably contain a lot of useful information to support users in the algorithm selection problem. However OpenML lacks of tools to support algorithm recommendation.

1.1 Motivation

The algorithm selection problem was identified by [Ric75] almost 40 years ago.

When the problem is a data mining task, the selection becomes an even more difficult task. Running several learning algorithms on large datasets in order to identify which one may have the best results is not only time but resource consuming. Therefore a solution that can help users (novices and/or experts) to identify the most promising algorithm for a given dataset is much needed.

Different attempts have been made to approach the problem of selecting the most suitable algorithm to use on a given dataset. Although some of them have shown very interesting results, some of the main challenges in algorithm selection for data mining still remains unsolved. The need of background knowledge, lack of user guidance, simplicity, process transparency, lack knowledge sharing and tool restriction, to name a few.

In the most common scenario, a user has to download a data mining program (Sections 2.2.1.1, 2.2.1.2), learn how to use it, understand how to implement each algorithm and run the experiment itself. Lastly the user has to comprehend the results achieved and be able to answer the following questions **"Which algorithms should I execute?"**, **"Was the result satisfactory?"**, **"Can I improve this result?"** and **"Is the cost of executing a new experiment worth it when compared to the possible gain it might bring?"**.

Another main limitation in data mining tools is the lack of knowledge sharing among experiments. After each experiment, the knowledge generated in each execution will only be available to the user, or a small group, that executed the task, making its performance only relevant throughout a short window of time when not explicitly shared.

Finally, even if all the previous questions have had a satisfactory answer another one spurs, **"How to use such information?"**. Gathering knowledge and experiments may already seem a difficult yet necessary task, but its the *how* to use such information that will determine the importance of the gathered knowledge.

[1] http://www.myexperiment.org/
[2] http://openml.org

1.2 Main Objective

In this project we will try to address two of the identified issues, **"How to use such information?"** and **"Which algorithms should I execute?"**.

There are already some tools that have addressed these issues, one of them is DM Advisor. DM Advisor was an online data mining tool for algorithm recommendation. It also introduced an interesting approach of a distributed execution strategy. However, the tool presented some issues with scalability and did not took full advantage of its distributed approach.

The main objective of this dissertation is to improve/rebuild the tool DM Advisor (Section 2.2.2.6) and integrate it with OpenML (Section 2.2.2.1) while trying to respond the two issues. Allowing users to submit a dataset, retrieve a ranking of the most promising algorithms, and ultimately execute some of the recommended algorithms on the uploaded dataset.

By providing such platform firstly we will be responding, partially, to Rice's algorithm recommendation problem. Secondly the platform will reduce the end-user from tool dependency and background knowledge. Another advantage, is the integration with OpenML. By using its data to learn our meta-model, it will be able to learn not only from the platform own experiments but from all the experiments that are and will be present in the OpenML database. Another advantage of our approach is because the experiments will be ran in a distributed way (delegating each experiment to its correspondent machine), the platform will be able to run multiple executions at the same time, making knowledge-gathering faster and increase its scalability. We will be able to store the experiment results in an automatic way, thus reducing potential knowledge loss when the experiment meta-data is not explicitly shared. Lastly, it will also have the ability of sharing data with others through OpenML.

In conclusion the goal of this project is to create an Automated Distributed Intelligent Discovery Assistant (ADIDA).

1.3 Overview

Besides the introduction, this report contains 4 more chapters. In the next chapter (2), we will introduce the main ideas related to the field of meta-learning and present some related works. In chapter 3, we will present our solution perspective, its architecture and implementation. In Chapter 4, we will explain some of the main use cases. Finally, in chapter 5 we will present our final remarks and talk about a few ideas for future work that have spurred during the dissertation.

Chapter 2

State of the art

> "(...) for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class" [WM97].

An Intelligent Discovery Assistant (IDA) is a tool that helps data miners exploring the space of valid data mining processes. Several attempts have been made at developing IDAs, some examples are [AC98], [BPH05] and [Eng96]. Two main problems with the research in the field of IDAs are: its distribution over many disciplines (statistics, exploratory data analysis, data mining, planning); and they address different kinds of support situations, such as support for the whole KDD process, among others. Furthermore, the IDAs developed up to now rely on many types of different background knowledge [SVB12].

2.1 Algorithm Recommendation and Meta-learning

Data mining is a process widely used to aid in decision making. Its concept will be explained in the following section.

2.1.1 Data Mining

Data mining is "*(...) the non-trivial extraction of previously unknown and potentially useful information from data*" [FPSS96a]. It is the analysis step of a Knowledge Discovery in Databases (KDD) process, and an interdisciplinary sub-field of computer science. It combines artificial intelligence, machine learning, statistics and database systems [CEF+06].

Data mining is usually an automatic, or semi-automatic task, that analyses large quantities of data to extract previously unknown information and interesting patterns. Such patterns can then be seen as a summary, and may be used in further analysis. For example, identifying multiple groups in the data to obtain more accurate prediction results by a decision support system. Some

of the most notable uses of data mining are games ([CG05], [WM09]), business ([Ver09]), science ([HG08]), engineering ([HG08], [XTLL09]) and medical ([WBK06], [CW02]), to name a few.

The goal of data mining is to extract information from a dataset, or past data, and transform it into an understandable structure. Discovering information from data takes two major forms: description and prediction [MR10]. Common types of data mining analysis include anomaly detection, association learning, cluster detection, classification and regression. The utilization of each of these data mining types provides a different perspective over the collected information, so it is pertinent to consider a basic review of each:

- Anomaly detection: in a large dataset it is possible to get a picture of what the data tends to look like in a typical case. Statistics can then be used to determine if something is notably different from the regular pattern. For instance, one could model tax returns and use anomaly detection to identify specific returns for review and audit.

- Association learning: this is the type of data mining that drives recommendation systems. For instance, this might reveal that customers who bought milk and diapers also often buy beer. These types of findings are often used for targeting and advertising.

- Cluster detection: a type of pattern recognition useful in recognizing distinct sub-categories (clusters) within the data. This method captures the relevant distinctions between similar groups in the data, while letting the data itself determine the groups. It uses algorithms to detect all of the different subgroups within a dataset that differ significantly from each other. In a simple example, we can imagine that the purchasing habits of different hobbyists would look quite different from each other: gardeners, fishermen and soccer enthusiasts would all be quite distinct.

- Classification: can be used to classify new cases into pre-determined categories. Learning from a large set of pre-classified examples, algorithms can detect systematic differences between items in each group and apply these corresponding models to new classification problems. Spam filters are a great example with a high degree of accuracy.

- Regression: a technique used to fit an equation to a dataset. It uses formulas to determine appropriate values to predict an output based on a given input. For example, to predict a relationship, factors like the amount of information, number of photos, friends, comments, likes, etc., could all be included in a model. Over time, this model could be tweaked to weight things differently. Such findings, could then be used to guide a design in order to encourage more of the same behaviours.

A data mining process can provide different results depending on the specific algorithm used, and understanding each possible combination and results makes *algorithm selection* a non-trivial process. Although, for the experienced *miner*, this huge amount of choices may prove itself necessary, it makes novice data miners usually overwhelmed by the plethora of algorithms available. Additionally, the performance of an algorithm depends greatly on the characteristics of the data.

There is no single algorithm that works best on all given problems [Wol95]. The wider the offer, the more difficult it becomes determining a suitable algorithm. Given a predictive task like classification, one will find a considerable amount of available classifiers. Some examples are Neural Networks, Support Vector Machines, k-Nearest Neighbours, Naive Bayes, Decision Trees, among others. The successful application of any of these algorithms, depends on the understanding of the algorithm itself, its applications and possible uses. Take for example k-Nearest Neighbours, a data mining algorithm who was used in this dissertation.

K-NN or k-nearest neighbours, is a non-parametric method for classification and regression [Alt92], and is one of the simplest algorithms of machine learning.

In k-NN the similarity between examples is usually based on a distance measure or by using simple rules like the *Euclidean distance*. The algorithm can be described in two steps:

1. Select a set of training examples containing the ones that are most similar to the new example in terms of their description.

2. Combine the target values of all selected examples to generated the prediction for the new example.

If the method is used for regression, it assigns the value of the object to be the average of the values of its k nearest neighbours. In this method, weights can be given to the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones.

When used for classification, it predicts objects values based on the k closest examples in the feature space. In other words, an object is assigned to the class most common amongst its k nearest neighbours.

Since the k-NN algorithm depends on the types of features that are used and not on the type of task the distance functions that can be used can be any common distance measure, such as weighted or unweighed Euclidean distance. For example:

$$distance(i,j) = \sum_{p=1}^{m} \frac{\left\| x_{i,p} - x_{j,p} \right\|}{max_l(x_{l,p}) - min_l(x_{l,p})} \tag{2.1}$$

Were x_i is the meta-feature vector of the meta-example i and m is the number of meta-features. The distance value for each meta-feature is normalized by dividing it by the corresponding range of values [BGCSV08].

To summarize, a figural representation of the data mining multiple categories complexity can be seen in figure 2.1.

2.1.2 Meta-learning for algorithm selection

In data mining the application of machine learning has spurred the research community in a research direction known as meta-learning. Meta is a prefix used in English to indicate a concept

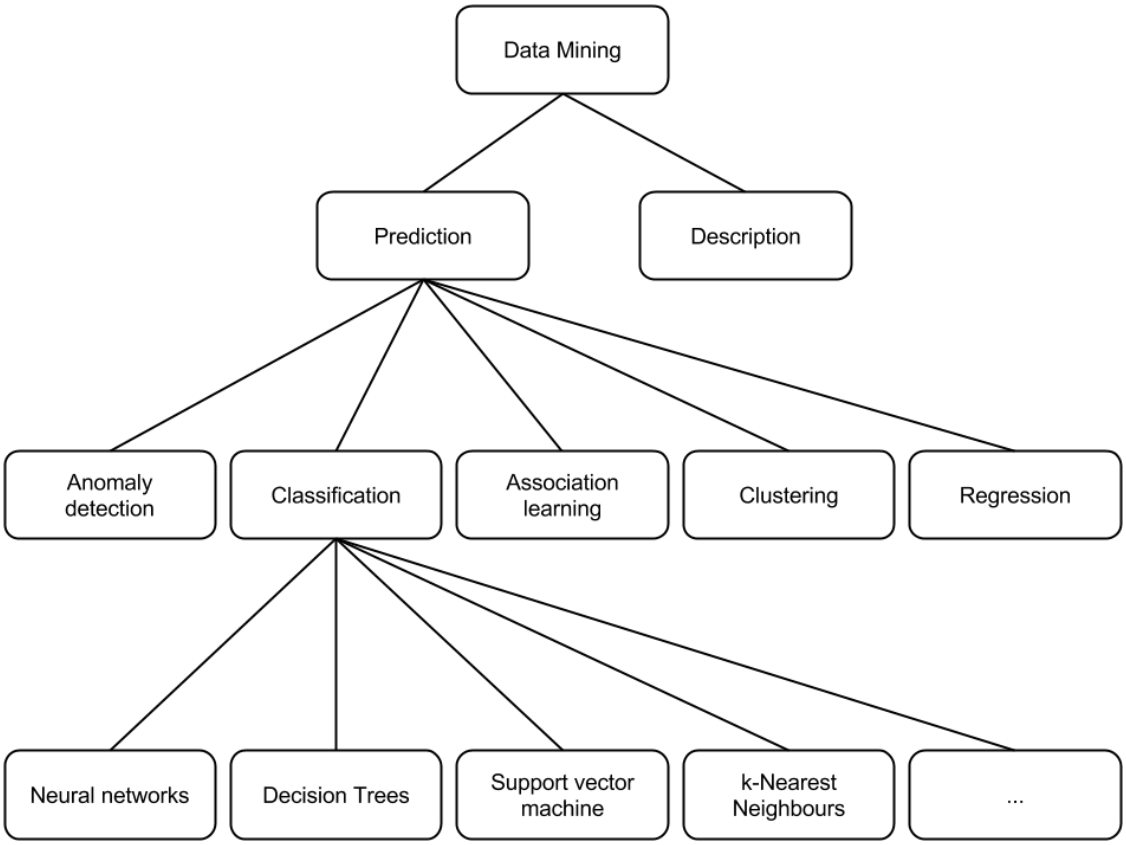

Figure 2.1: Data mining categories and sub-categories

which is an abstraction from another concept. So for example, in computer science meta-data is data that describes other data (data about data).

Although several researchers may hold different views of what meta-learning exactly means, it may be define as *"(...) the study of principled methods that exploits meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes."* [BGCSV08].

Meta-learning studies how learning systems can become more effective through experience. Not simply how a good solution can be found but that this be done increasingly more effectively through time. A unifying point in meta-learning lies in how to exploit knowledge acquired on past learning tasks to improve the performance of learning algorithms. It differs from *base* learning in the level of adaptation. Learning at a *base* level is focused on accumulating experience on a specific learning task. Learning at a meta level, however, is to accumulate experience on the performance of multiple applications of a learning system, or in other words *learning to learn* [BGCSV08].

A *base* learner will produce an hypothesis that normally improves with an increasing number of examples, however successive applications of the learner on the same data, under the same experimental conditions, will always produce the same hypothesis, independently of performance. In other words, no knowledge is extracted across domains or tasks [BGCSV08].

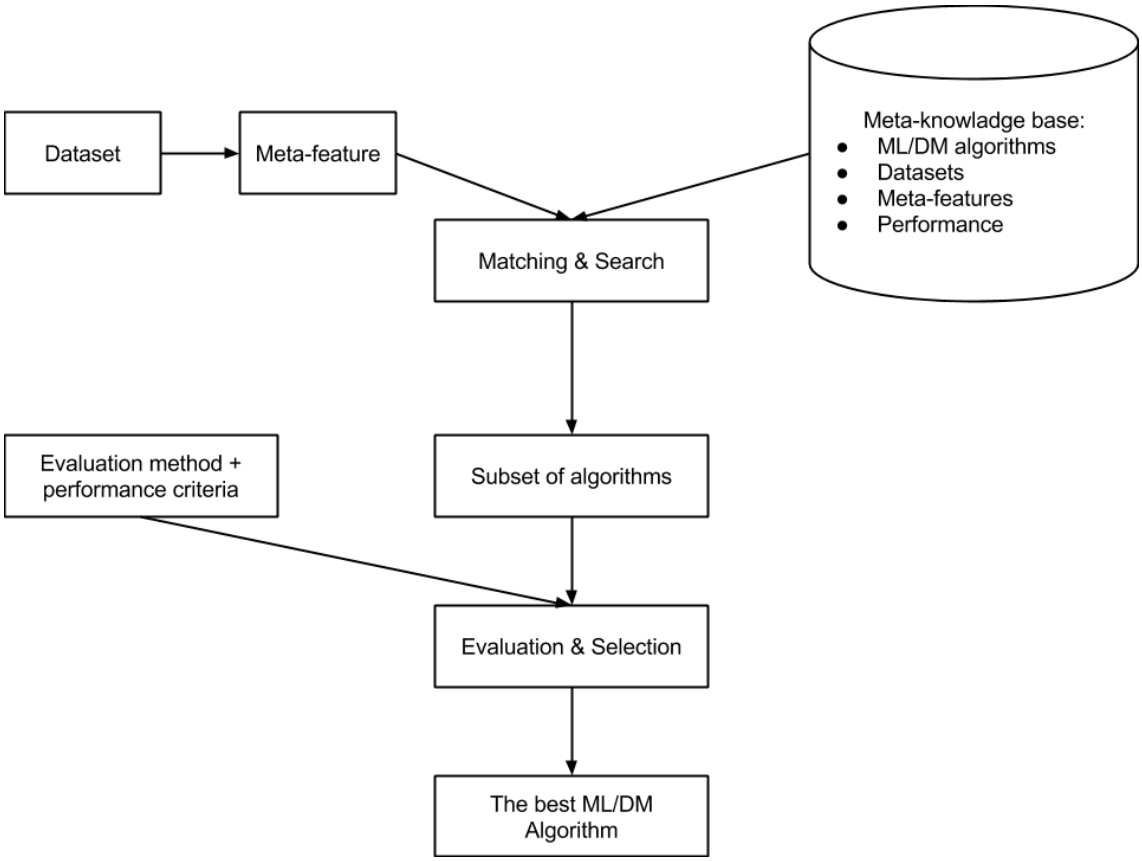

Figure 2.2: Selection of ML/DM algorithms: finding a reduced space and selecting the best learning algorithm. Source: [BGCSV08]

The key to solving this problem is gathering knowledge about the learning process (meta-knowledge). This knowledge is built based on data that describes the performance of algorithms and the characteristics of the problems (meta-data). Such knowledge can then be used to improve the learning mechanism itself. A diagramatic representation (Figure 2.2) of this process was presented by [BGCSV08].

The development of meta-learning systems for algorithm recommendation involves addressing several issues not only at the meta level (lower part of figure 2.3) but also at the *base* level (top part op figure 2.3). At the meta level, it is necessary to choose the form of the recommendation that is provided to the user, i.e., the type of the meta-target feature. This type will determine the type of the meta-algorithm, which are the meta-learning methods that can be used, which in turn determines the type of meta-knowledge that can be obtained. Such process will generate meta-knowledge that relates the properties of those datasets to the relative performance of algorithms, so it is necessary to define which properties are important to characterize those datasets and to develop meta-features that represent those properties [BGCSV08].

The information stored on the meta-database must also contain the performance of the *base* algorithms on the dataset and the measures that will be used to evaluate the performance. Careful

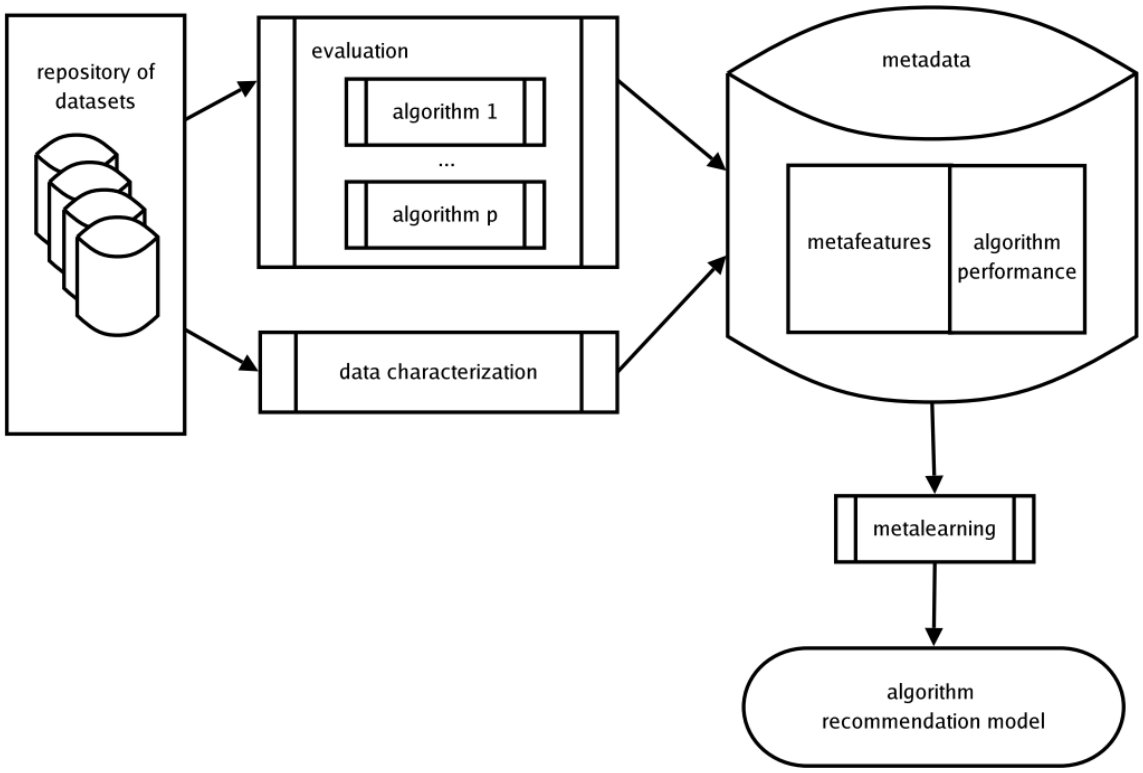

Figure 2.3: Meta-learning to obtain meta-knowledge for algorithm selection. Source: [BGCSV08]

selection of the base-algorithms, measures of performance evaluation and performance methodology estimation must be made, so that the values can be reliable and comparable [BGCSV08].

Several European research projects like StatLog [FMST94], ML Toolbox [SRC+95], METAL [GCFP+] and MiningMart [MS04] have addressed the algorithm recommendation problem each of them contributed with important advances [BGCSV08].

The STATLOG Project did comparative studies of different machine learning, neural and statistical classification algorithms. About 20 different algorithms were evaluated on more than 20 different datasets. The results were published in several papers and in the book [MST94]. It tried to explain the performance of the algorithms according to the dataset properties. They automated the process by constructing meta-learning problems, and characterized each algorithm for each dataset. The project however had two major issues. It only used statistical values from the datasets properties. Secondly, the algorithms were only characterized as applicable or non-applicable [KH01].

The Machine Learning Toolbox is an European project aimed at the exploitation of ML techniques in "real world" industrial and medical domains. Several principal activities were pursued within the project, such as building an integrated toolbox that provided a wide range of ML techniques, investigating the applicability and use of ML techniques on a number of selected industrial problems (Case Studies) and producing an advisory system. ML Toolbox also provided a number of essential functions for machine learning, especially for data clustering and pattern recognition.

However, detailed information about the project is scarce.

MetaL [GCFP+] aimed at the development of methods and tools for providing support to users of machine learning and data mining systems. Started in 2001, the central goal of METAL was to develop a prototype assistant system that supported users with model selection and method combination, and guiding them through the space of experiments. The system would combine prior meta-knowledge with meta-level learning. For each constituent, known techniques would be consolidated and original ones would be developed to cope with novel learning situations and applications. The meta-knowledge base of the assistant, which integrated expert-given meta-information and the results of meta-learning, provided key information about past usage of ML systems. This knowledge described the conditions under which operations carried out in the past had succeeded or failed. The proposed system aimed to exploit this information while providing guidance to the user. Such guidance would not be restricted to selection of an appropriate method only, but would also suggest data transformation steps that are often crucial to obtain good results. Furthermore, the systems guidance would not be limited to a single choice. It would consist of a set of promising operations that the user may readily explore in an orderly fashion. The system would extend its meta-knowledge base dynamically as it was used, and hence have the capability to adapt to specific environments. Metal provided a ranking which predicted how well a set of algorithms would perform on a particular dataset. The ranking was generated using knowledge of the performance of those algorithms on some benchmark datasets. The ranker first characterized the previously unseen dataset and based on this characterisation, and on a similar characterisation of the benchmark datasets, datasets were selected which are similar to the unseen dataset. Metal would then use its knowledge of the performance of the algorithms on the benchmark datasets to predict the performance of the algorithms on the unseen dataset.

MiningMart ([MS04]) aims at developing new techniques that support user-guided representation adjustment, as well as techniques that automatically select or change representations. The basic idea behind MiningMart is to store best practice cases of preprocessing chains that where developed by experienced users. The data is described on the meta level and is presented in application terms. MiningMart users choose a case and apply the corresponding transformation and learning chain to their application.

In predictive tasks it is possible to evaluate models quantitatively. Therefore given a dataset it is possible to objectively compare two algorithms. In this context, algorithm recommendation is possible because you can choose a best algorithm for a problem. In descriptive data mining, evaluation is typically subjective and thus it is harder to define an algorithm recommendation problem.

2.1.2.1 Spearman Rank Correlation Coefficient

The Spearman's rank correlation coefficient ([Spe04]) is a non-parametric[1] measure of statistical dependence between two variables, appropriate for continuous and discrete variables, including

[1] Does not assume that the data has any characteristic structure or parameters.

ordinal variables. It assesses how well the relationship between two variables can be described using a monotonic function[2]. If there are no repeated values, a perfect Spearman correlation of +1 or −1 occurs when each of the variables is a perfect monotone function of the other. When used to assess how similar two rankings are, it enables the evaluation of the accuracy of a given ranking. For instance, given the target ranking (1, 2, 3, ..., n − 1, n), if ranking A generates the ordering (2, 1, 3, ..., n − 1, n) and ranking B generates the ordering (n, n − 1, ..., 3, 2, 1), A will be considered more accurate than ranking B. This is because the ordering of A is more similar to the target ranking than the ordering of B.

$$rs = 1 - \frac{6\sum_{i=1}^{n}(P_i - T_i)^2}{n^3 - n} \tag{2.2}$$

2.2 Data Mining and Algorithm Recommendation Tools

We will now present a brief summary of some commonly know data mining tools. Followed by the presentation of some projects who have address the algorithm recommendation problem.

2.2.1 Data Mining Tools

A data mining tool is a powerful software that makes use of data mining algorithms, and supports a complete KDD process [MR11]. In the following sections we will briefly address some of the most commonly used tools, namely, RapidMiner, Weka and R.

2.2.1.1 Rapidminer

RapidMiner[3] (Figure 2.4) is an open-source system for data mining. It is available as a stand-alone application for machine learning, text mining, predictive analytics, business analytics and data analysis. It is used for research, education, training, rapid prototyping, application development, and industrial applications and recently a number of algorithm recommendation tools (2.2.2.4, 2.2.2.3, 2.2.2.5) have been integrated into it through plug-ins[4].

2.2.1.2 Weka

Weka is a popular suite of machine learning software written in Java, with a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from other Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

It supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. It also provides access to SQL

[2] A function between ordered sets that preserves the given order.
[3] http://rapidminer.com
[4] Piece of software that enhances a software application. Usually cannot be run independently.

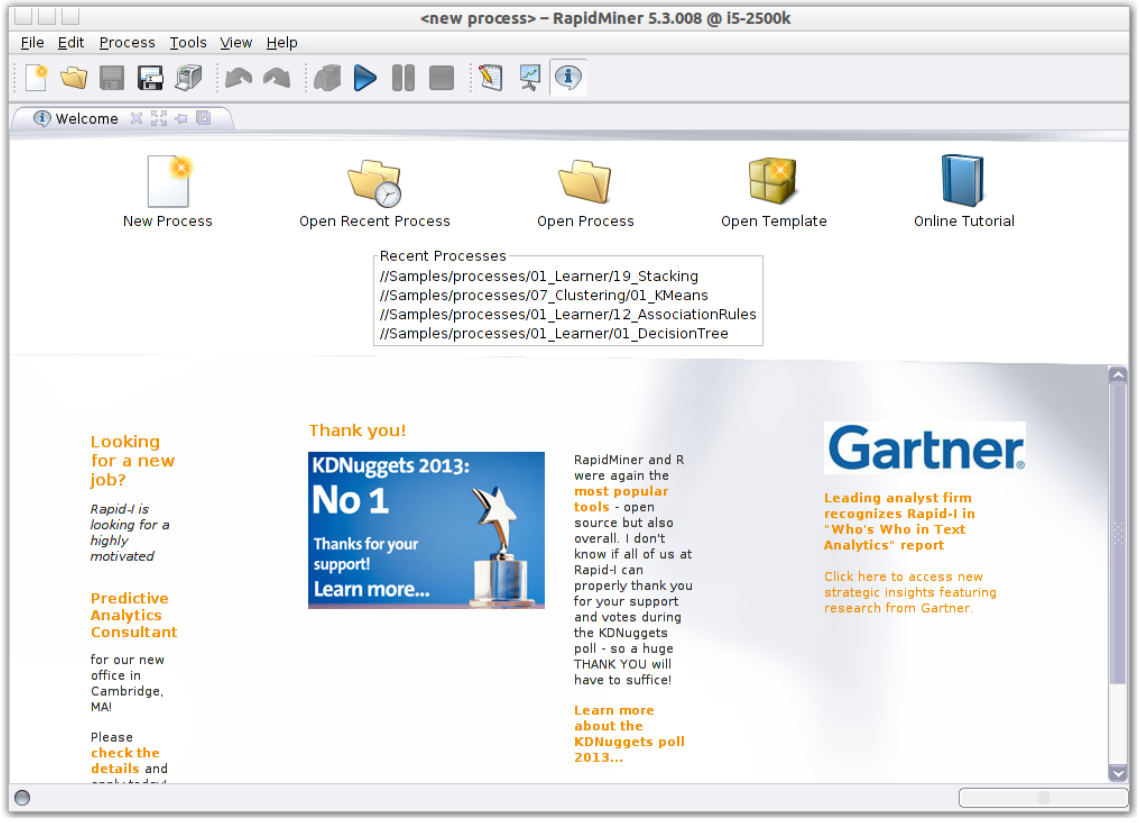

Figure 2.4: Rapidminer

databases using Java Database Connectivity and can process the result returned by a database query.

Its development started back in 1997 and is now used in many areas, particularly in education and research.

Figure 2.5: Weka: GUI

2.2.1.3 R language

Created by Ross Ihaka and Robert Gentleman [IG93], R is a programming language and software environment for statistical computing and graphics. Widely used among statisticians and data miners to develop statistical software [FA05], R popularity has increased substantially in recent years [Mue].

R is an implementation of the programming language S ([Gun02]) combined with lexical scoping semantics inspired by Scheme ([RC86]). Written primarily in C ([KR88]), R is available on various operating systems and uses a command line interface. However, several graphical user interfaces have been made available for use (eg. [Fox05], [LL10]).

Powered with a wide variety of statistical and graphical techniques, including linear and non-linear modelling, classical statistical tests, time-series analysis, classification, clustering, and others. R can be easily extensible through functions and community packages due to its lexical scoping rules. However, many of R standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made.

R has stronger object-oriented programming facilities than most statistical computing languages and allows the direct manipulation of its objects from other programming languages, like C or Java ([GJSB05]), by linking and calling it at run time. Another strength of R is static graphics, which can produce publication-quality graphs and mathematical symbols. Dynamic and interactive graphics can also be produced, however they are only available through additional packages.

2.2.2 Algorithm Selection Tools

As seen earlier, the algorithm selection problem is one of the oldest problems in computer science. In the following sections, some of the projects that have addressed this issue and their respective approaches will be presented.

2.2.2.1 OpenML

OpenML[5] is an open science platform for machine learning that allows the submission of datasets, algorithms, workflows, experiments and results. It automatically organizes all content in a database, where it is freely available to everyone and searchable through their website.

In OpenML experiments are defined as *tasks*. This is done so that experiments from different researchers are comparable. A *task* is a well defined problem to be solved and must have the inputs provided and the expected output defined. Currently only supervised classification and supervised regression tasks are supported, but OpenML is designed to be extended to other task types.

The database contains all details about all the shared datasets and experiments. All the information in the database is openly available to everyone and all implementations, datasets or evaluation metrics are required to include meta-data (name, description, installation, etc.). A representation of the OpenML schema can be seen in figure 2.6.

The submission of new content is also possible through its RESTful API or using the available plug-ins that are being integrated with various popular data mining/machine learning tools like RapidMiner (Section 2.2.1.1) or Weka(Section 2.2.1.2). To explore OpenML various search interfaces are also available, such as web interface textual search or SQL statements and the result of the queries can be obtained in CSV or ARFF.

2.2.2.2 Weka Metal

WekaMetal is a Meta-Learning extension to the data-mining package Weka (Section 2.2.1.2). It uses adaptations of the algorithms developed by the MetaL consortium to provide advice on algorithm selection, based on expected accuracy and time performance.

WekaMetal was designed to work with Weka 3.2, this version however dates back to 2002, since then no further versions or developments have been made so it is probable that the project has stopped.

2.2.2.3 ML Wizard

ML Wizard is a RapidMiner (Section 2.2.1.1) extension, where for a given dataset (Figure 2.8) it automatically recommends a classification process based on certain characteristics of a dataset, by extracting meta-features from the dataset (Figure 2.9) and predicting accuracies (Figure 2.10) for certain classifiers along with a Root Mean Squared Error (RMSE) value. A small RMSE value indicates higher confidence in the prediction. The user can then select which classifiers he wants

[5] http://openml.org

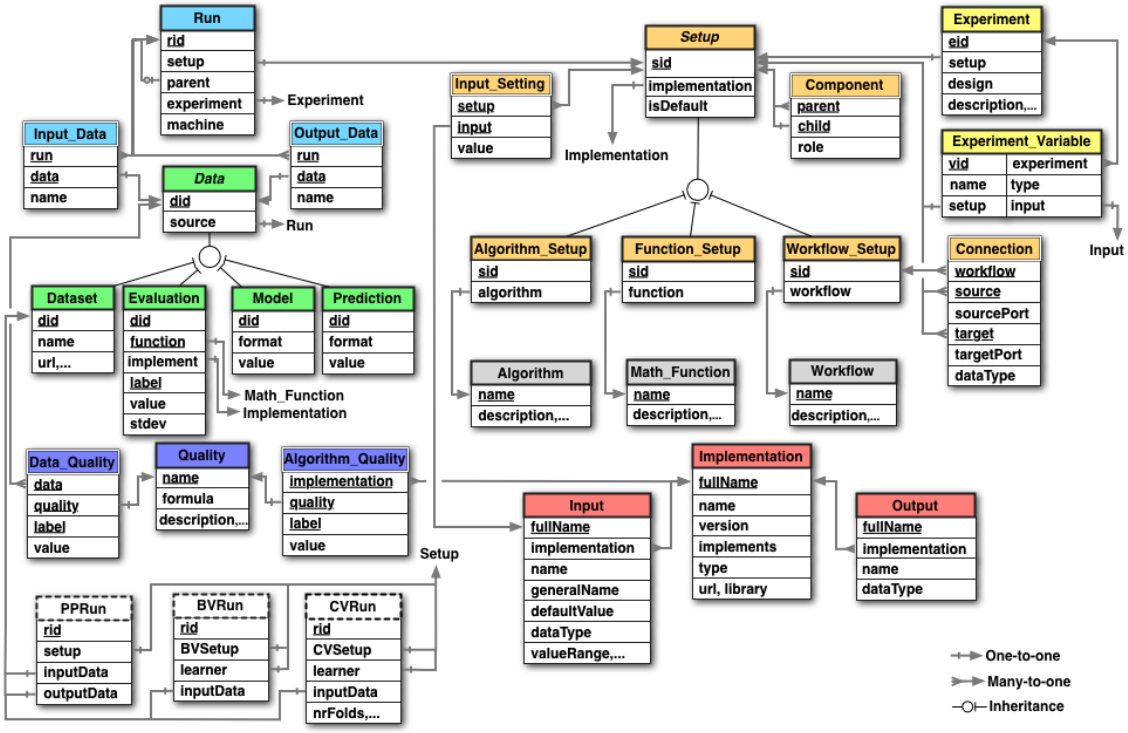

Figure 2.6: OpenML database schema. Source: http://www.openml.org

to evaluate (Figure 2.11) in more detail to get an optimal parametrization (Figure 2.12). Finally the user can choose to load the generated process (Figure 2.13) from one of the chosen algorithms he choose earlier to be executed.

ML Wizard is a very simple tool to use and its functionality is very straightforward. However there are two main issues with this plug-in: it is tool-dependent, if the user does not have access to the RapidMiner tool, he will not be able to use the plug-in; the process of meta-learning is not open, there is not a clear explanation on how the algorithm recommendation is performed, the source of the meta-data was extracted, and additionally, the meta-data generated by the use of the tool is not shared. Additionally, it is also not very clear how the RMSE value is calculated. Usually an error can only be computed by comparing a prediction with a true value.

2.2.2.4 Data Mining Assistant

The DM Assistant is a RapidMiner (Section 2.2.1.1) plug-in that helps create processes by recommending operators that fit well with the operators already contained in the current process. The current process is sent to a Web service (Figure 2.14) which computes recommendations similar to the ones well known for books: "*People who like to use X-Validation also use Model-Y*".

Whenever the processes is changed in RapidMiner, the current process is sent to a Web service as a query. The Web service computes appropriate recommendations and returns a list of operators

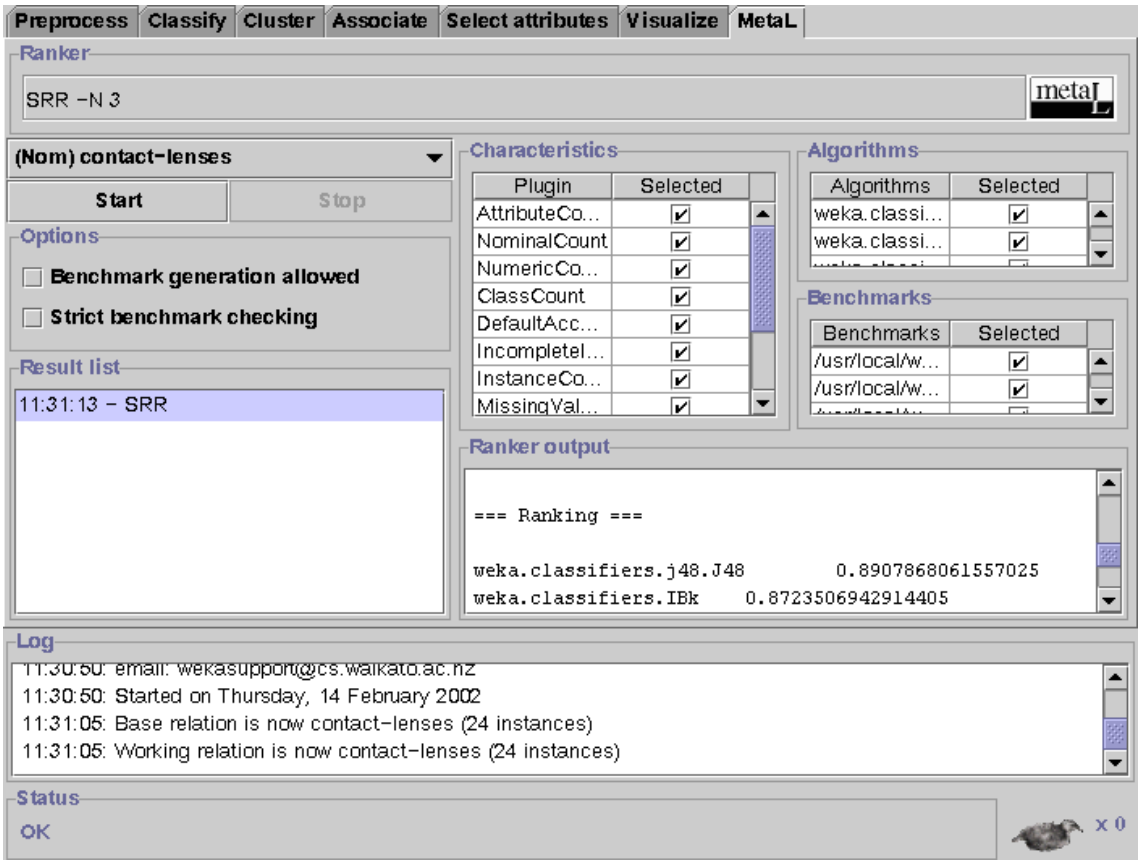

Figure 2.7: WekaMetaL (version 0.11)

(Figure 2.15). This Web service can be trained automatically over time by analysing the processes it sees. Thus it constantly improves its recommendations.

As the ML Wizard (section 2.2.2.3), the Data Mining Assistant is specific for the RapidMiner (section 2.2.1.1) tool. Additionally if the user does not have any kind of background knowledge, this process, although it seems simple, it may guide the inexperienced user to an uncertain work flow, giving unexpected results in the end. Additionally, it is also unclear how the recommendation are generated, namely if there is any kind of learning behind the system.

2.2.2.5 Intelligent Discovery Assistant

The Intelligent Discovery Assistant, another RapidMiner (Section 2.2.1.1) plug-in, helps the user creating data mining processes. Based on the specification of input data and a modelling task, it automatically creates processes tailored specifically to the data. Based on the analysis of hundreds of processes, it selects operators that are specifically well-suited for the problem and data set at hand. It chooses operators that have achieved good accuracy on similar data sets in the past. Furthermore, it takes care of the preprocessing which may be necessary for applying certain algorithms. Here, too, appropriate preprocessing operators are selected based on their projected impact on the overall performance of the process.

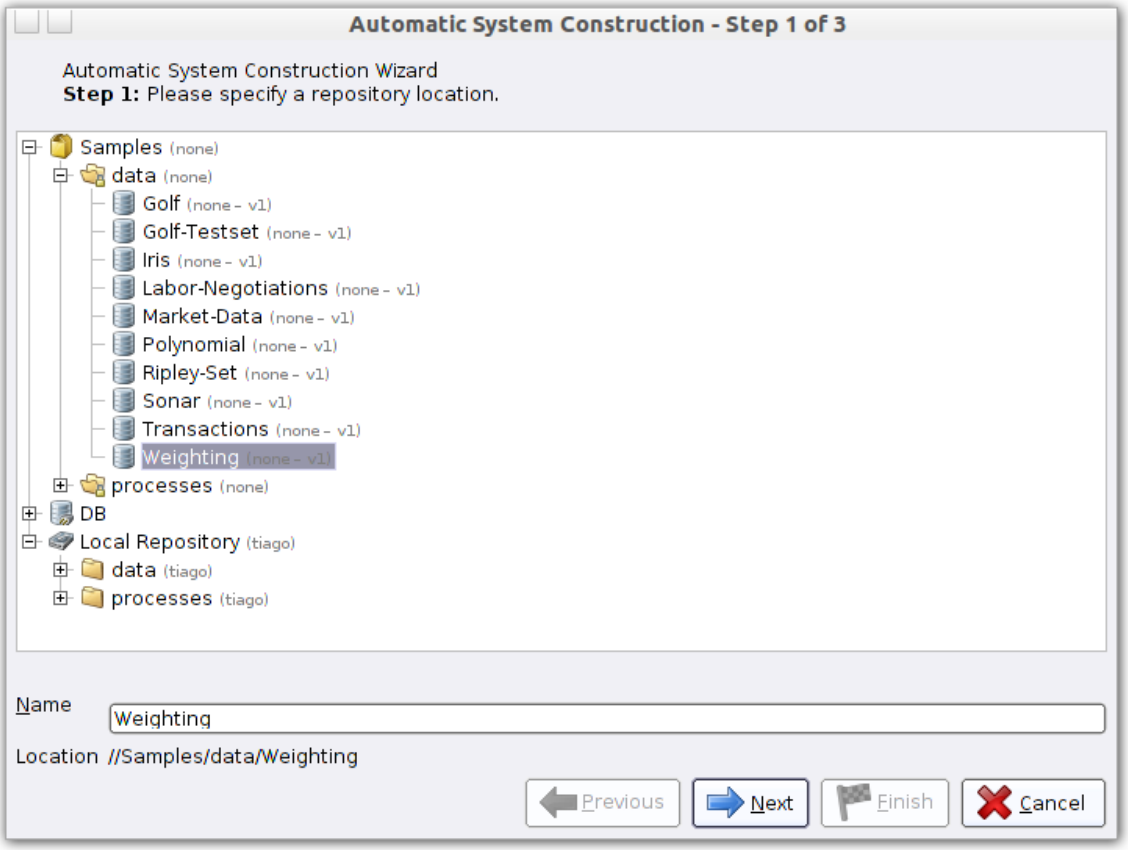

Figure 2.8: ML Wizard: Selecting a dataset.

Creating a data mining process using the Intelligent Discovery Assistant can be done by following four simple steps:

1. Import data. If your data is already saved in your RapidMiner repository or on RapidAnalytics, there is nothing to do. Otherwise, you can import a CSV or Excel file into your repository.

2. Select data. You can choose up to two data sets by dragging them into the respective area. Once the data sets are dropped here, the meta-data will be displayed. The IDA will automatically identify appropriate roles: training data, test data, or application data.

3. Select goal. You can select the goal or task you want to execute. A short informational text is displayed on the right hand side, describing the individual options (See option 3 in Figure 2.16).

4. Evaluate work-flows. Once the above settings are defined, the instructions in Step 4 should vanish and you should be able to click on "Fetch plans". The actual planning can take a while. A number of work-flows will be generated and displayed in a table. You can select those which you consider promising and click "Evaluate" to execute these processes and

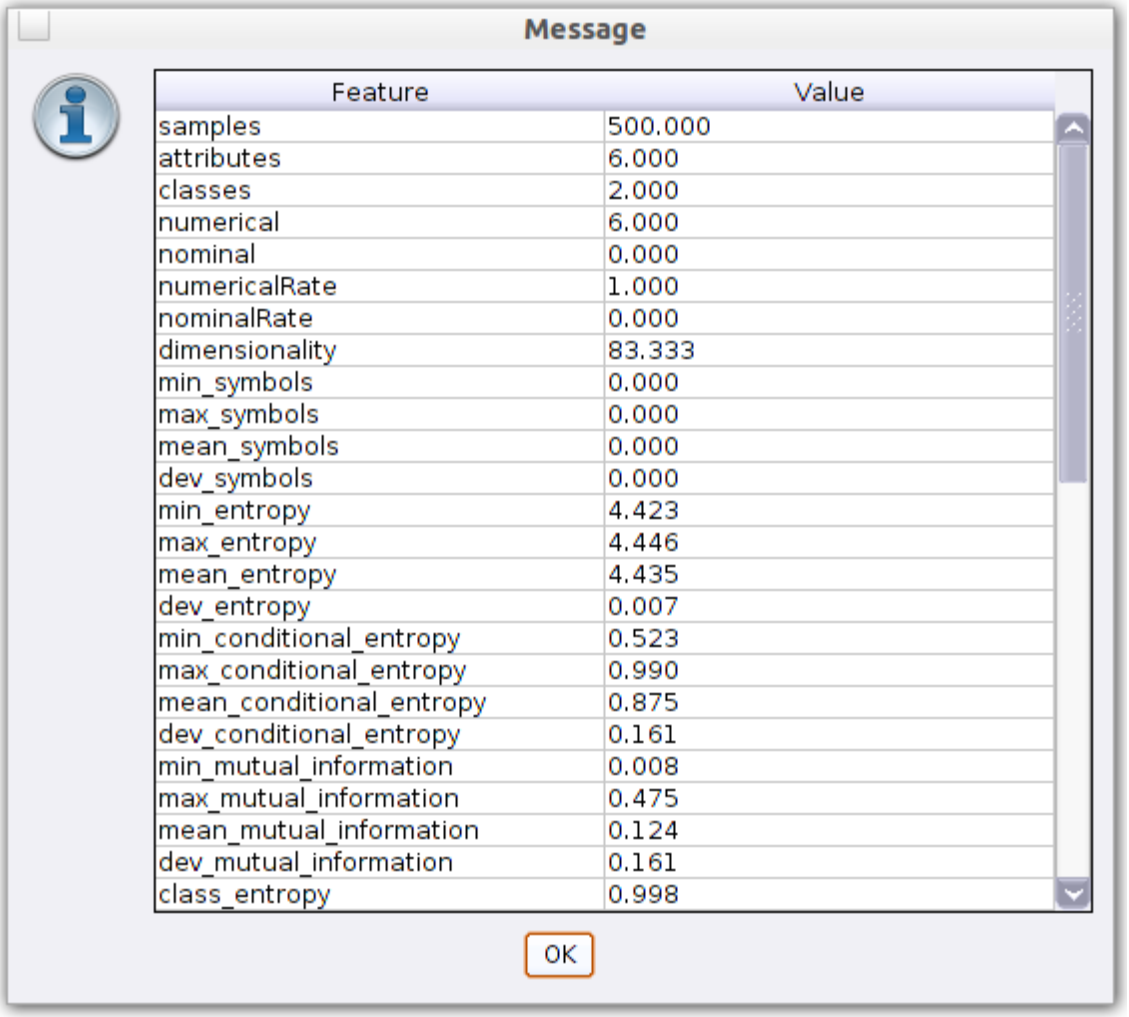

Figure 2.9: ML Wizard: Calculated meta-features.

see their respective performances. Finally, you can chose to open one of the processes in RapidMiner which will bring you back to the process design perspective, as in Figure 2.13.

The Intelligent Discovery Assistant, as the ML Wizard and the Data Mining Assistant (sections 2.2.2.3 and 2.2.2.4), suffers from the same issue, the plug-in needs the user to have some knowledge on using RapidMiner and installing plug-ins in it. The Intelligent Discovery Assistant uses meta-learning to select operators that are specifically well-suited for the problem from similar data sets already analysed. However, the user experiment is executed locally and the final and actual outcome of the experiment is not uploaded back to the database. So in the end meta-data (and potentially, meta-knowledge) is being lost.

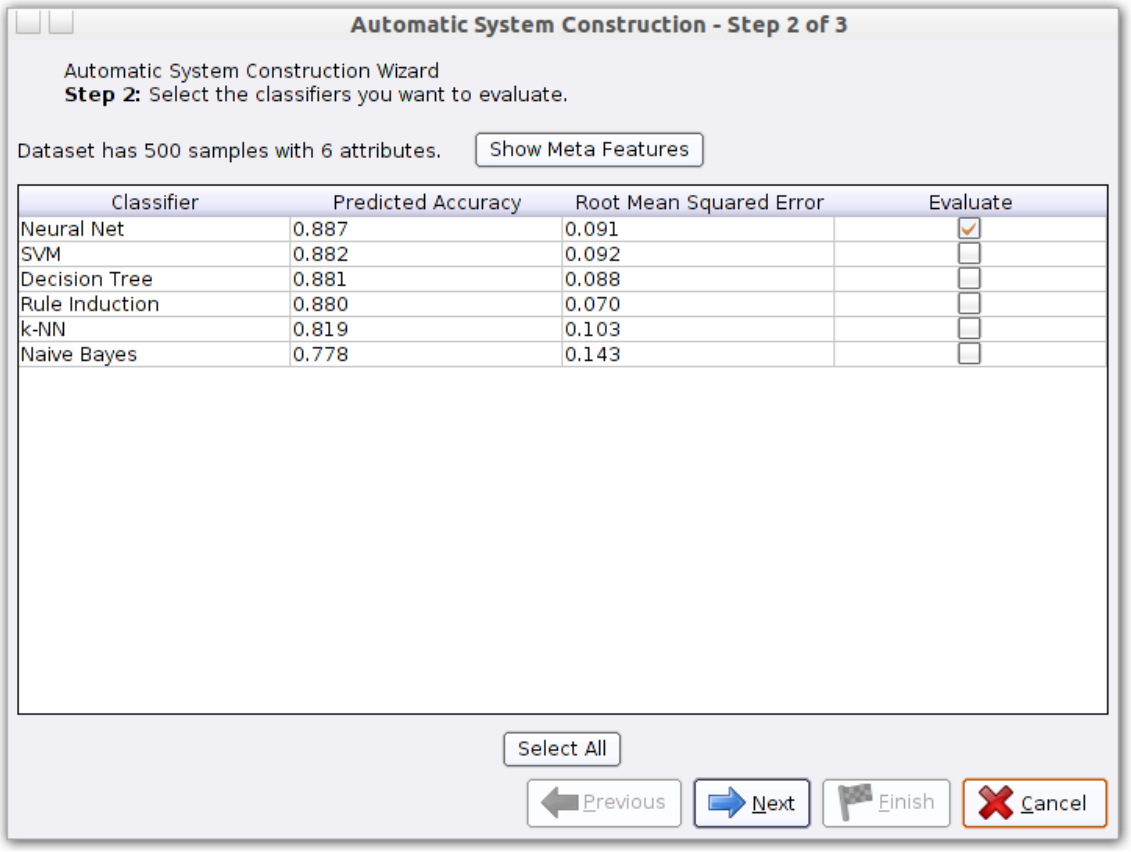

Figure 2.10: ML Wizard: Prediction.

2.2.2.6 Data Mining Advisor

Data Mining Advisor[6] (Figure 2.17) was an on-line data mining tool. It allowed the submission of a dataset to retrieve a ranking of algorithms. It also allowed the execution of some of the algorithms. However, scalability problems were detected since the project had a centralized execution, in other words, the algorithms were executed server side. A user could also upload a new algorithm and identify which type of problems it can solve. While this approach seamed feasible for a limited number of users, if the number increased considerably the system would not be able to respond properly, so a new version was implemented.

The new variation, the Rank Advisor[7], is a web-based algorithm recommendation system for optimization. This system provided recommendation concerning which meta-heuristic is best suited to solve Job-Shop problems. In this new version different machines are responsible for the processing of different algorithms and the server functions has a broker. If a user made an execution request, the server would redirect the request to the owner of that specific algorithm. If

[6] Accessible only through the internet archives at https://web.archive.org/web/20120323135835/http://www.metal-kdd.org

[7] http://rank.fep.up.pt/Advisor

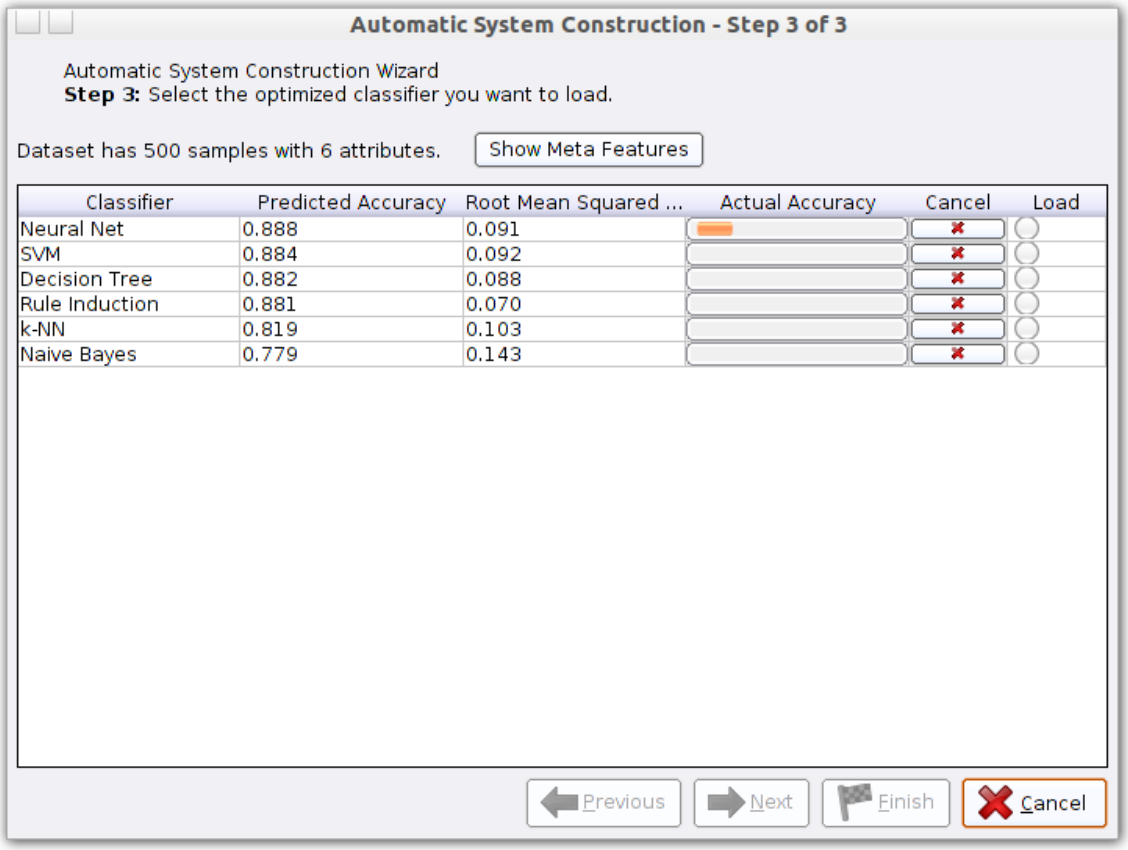

Figure 2.11: ML Wizard: Evaluating.

available the owner would executed the task. The results were then presented to the requesting user.

The Rank Advisor was an interesting approach for algorithm recommendation, focused on optimization problems. Their approach was indeed important, however the system had a limited database of experiments, limited algorithm availability and did not fully implemented meta-learning. The ranking system despite studied and somewhat tested was not embedded in the platform, so ultimately it did not took fully advantage of its possibilities nor the advantages of meta-learning. Furthermore, the platform did not implement the distributed algorithm recommendation architecture in a optimal way. It was the responsibility of the server to send the dataset, to request the execution and to retrieve the result. This gave the server excessive and unnecessary work. Additionally, while a dataset was being sent, the server was not able to reply to new requests until the previous task was being executed. Thus, it was not taking fully advantage of the distributed execution strategy.

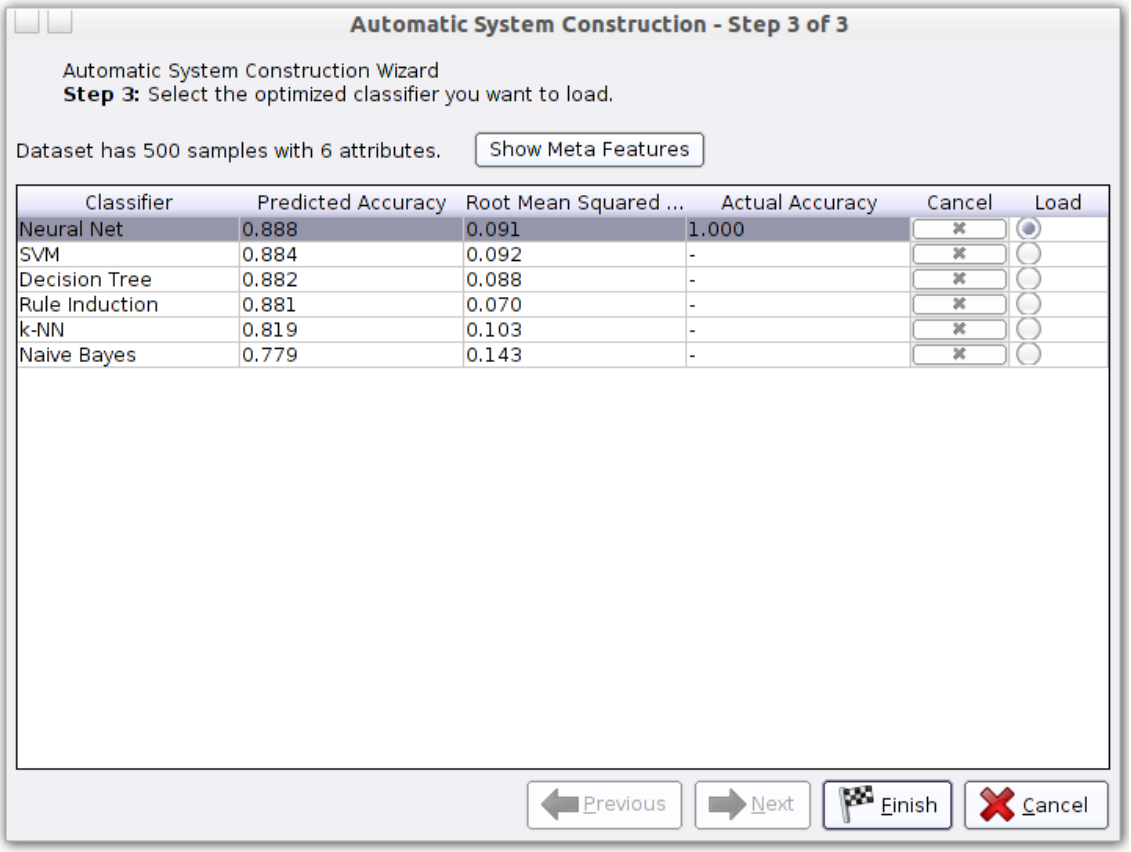

Figure 2.12: ML Wizard: Actual accuracy.

2.3 Web Development

Web development is the development of a web site, tool, or application for the Internet (World Wide Web). It is usually a collaborative effort between departments like web design, content development, client-side/server-side scripting, web server, network security configuration, among others.

2.3.1 Hypertext Transfer Protocol

HTTP is a protocol for distributed, collaborative and hypermedia information systems. It functions as a request-response protocol in the client-server computing model. A client submits a request message to the server and the server provides the resources, like HTML files or other content. The server can also perform other functions on behalf of the client, like returning a response message. The response contains a completion status information about the request and may also contain requested content in its message body.

HTTP is designed to allow that intermediate elements improve communications between clients and servers. For example, to allow websites with a high-traffic flow, the use of intermediate servers who deliver content on their behalf to improve the response time. Its resources are identified and

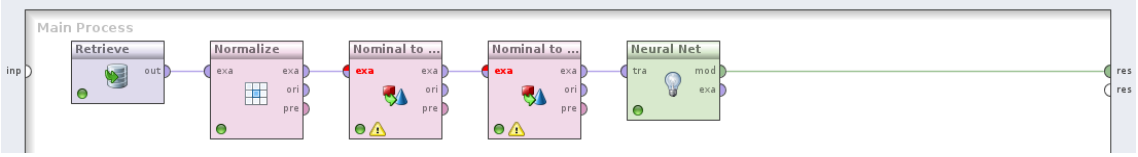

Figure 2.13: ML Wizard: Generated process.

located on the network by a Uniform Resource Identifier (URI) or, more specifically, Uniform Resource Locator (URL).

2.3.1.1 Request Methods

HTTP defines methods to indicate a desired action to be performed on the identified resource. In the 1.0 version, GET, POST and HEAD methods where defined [FN95]. In 1999 version 1.1 was presented, it added 5 new methods, the OPTIONS, PUT, DELETE, TRACE and CONNECT [FGM+99]. In 2010 PATCH was presented by [DS10]. Any server can be configured to support any combination of these methods and any client can access them. There is no limit to the number of methods that can be defined, allowing for future specifications to be added without breaking any existent infrastructure.

- **GET** Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect [FN95].
- **HEAD** Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content [FN95].
- **POST** Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data might be an annotation for existing resources, a message, a mailing list, a comment, an item to add to a database, among others [FN95].
- **PUT** Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an existing resource, it is modified, if not then the server can create the resource with that URI [FGM+99].
- **DELETE** Deletes the specified resource [FGM+99].
- **TRACE** Echoes back the received request so that a client can see what changes or additions have been made by intermediate servers [FGM+99].
- **OPTIONS** Returns the HTTP methods that the server supports for a specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource [FGM+99].

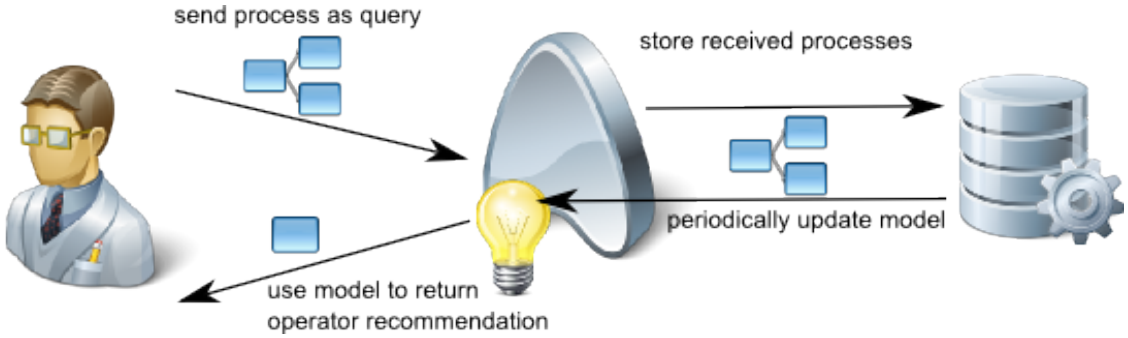

Figure 2.14: DM Assistant: the recommender Web service architecture. Source: http://www.e-lico.eu/dm-assistant.html

- **CONNECT** Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy [FGM+99].
- **PATCH** Is used to apply partial modifications to a resource [DS10].

HTTP servers are required to at least implement GET and HEAD methods.

2.3.1.2 Architectural Styles

The two most commonly used architectural styles are the Representational State Transfer (REST) and the Simple Object Access Protocol (SOAP).

2.3.1.2.1 SOAP Defines a standard communication protocol specification for XML-based message exchange. The protocol consists of three parts, an envelope which defines what is in the message and how to process it, a set of encoding rules for expressing instances of application-defined data-types and a convention for representing procedure calls and responses [BEK+00].

2.3.1.2.2 REST It describes a set of architectural principles by which data can be transmitted over a standardized interface. REST does not contain an additional messaging layer and focuses on design rules for creating stateless services. A client can access the resource using the unique URI and a representation of the resource is returned. With each new resource representation, the client is said to transfer state. While accessing RESTful resources with HTTP protocol, the URL of the resource serves as the resource identifier and GET, PUT, DELETE, POST and HEAD are the standard HTTP operations to be performed on that resource [Fie00].

2.3.1.2.3 REST vs SOAP SOAP is versatile enough to allow the use of different transport protocols, such as HTTP, SMTP, JMS and Message Queues. The SOAP model tunnels fine in the HTTP post/response model, so it can tunnel easily over existing firewalls and proxies without modifications to the SOAP protocol and can use the existing infrastructure. However, since the

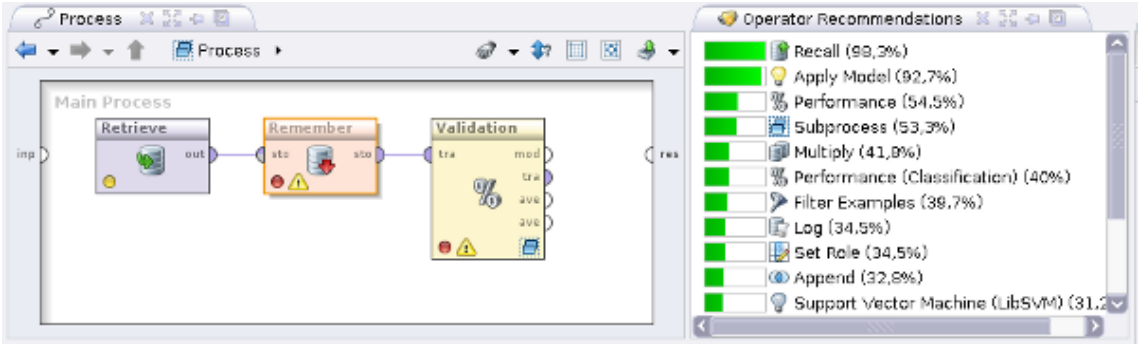

Figure 2.15: DM Assistant: Recommendations generated for the current process. Source: http://www.e-lico.eu/dm-assistant.html

Web 2.0 the trend has been on REST despite of SOAP, due to its more direct representational and resource-oriented architecture [BDS08].

2.3.2 Methodology

A software development methodology is a framework, or an approach, that is used to structure, plan, and control a development process. The methodology can include the development environment, the model-based development, and the use of a particular programming library and other tools.

2.3.2.1 Model-Driven Engineering

Model-Driven Engineering (MDE) is a software development methodology meant to increase productivity, compatibility between systems, simplifying processes and promoting communication between the teams working on the system. It focuses in on creating and exploiting domain models[8] rather than on the computing, or algorithmic, concepts.

Emerging in 2001 [OVK+03] it addressed the growing complexity of system architectures who lacked of integrated views and was forcing "*(...) many developers to implement suboptimal solutions that unnecessarily duplicate code, violate key architectural principles, and complicate system evolution and quality assurance*" [Sch06].

By combining domain-specific modelling (DSM) languages with transformation engines and generators, MDE creates source code, alternative model definitions and/or other artefacts, thus diminishing such issues. Also the ability to synthesize such artefacts helps to guarantee that the consistency between the application implementations and the analysis of the information associated with functionality and quality of the service requirements are captured by the models.

The usage of DSM languages ensures that the domain model is perfectly represented in syntax and semantics. Thus guaranteeing a smaller learning curve because the concepts of the language

[8] Abstract representations of the knowledge and activities that govern a particular application domain.

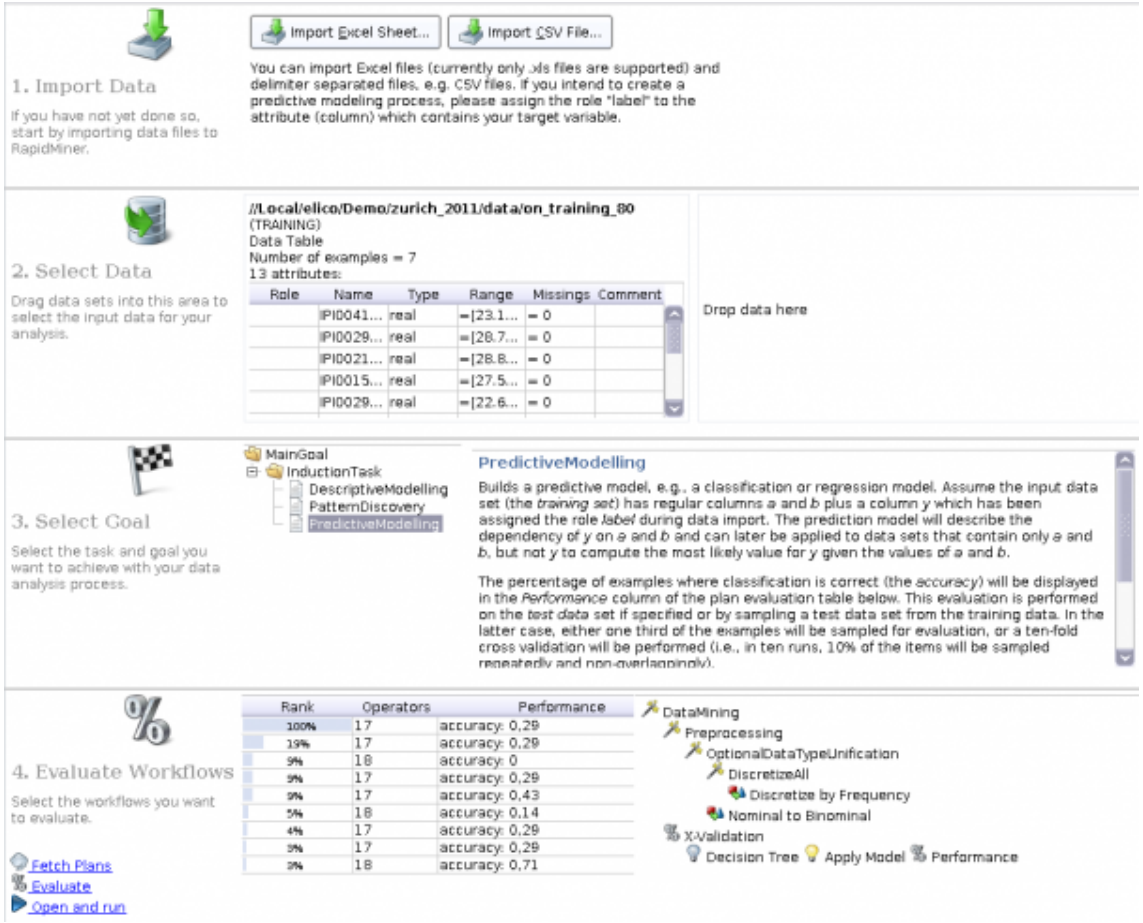

Figure 2.16: The four simple steps of the IDA view in RapidMiner

are already known by the domain experts. This also helps software architects, system engineers and other experts, to ensure that the software system meets the users needs.

2.3.3 Meta-Programming

> "A meta-programming system is a programming facility (sub-programming system or language) whose basic data objects include the programs and program fragments of some particular programming language, known as the target language of the system. Such systems are designed to facilitate the writing of meta-programs, that is, programs about programs. Meta-programs take as input programs and fragments in the target language, perform various operations on them, and possibly, generate modified target-language programs as output." [CI84],

Meta-programming is *code that generates code*. With the help of generators, high-level descriptions are decoded into low-level implementations. This allows developers to focus on specifications based on tested standards rather than implementation, making tasks like system maintenance and evolution much easier and affordable [Bas87, Cle88]. This paradigm and its advantages

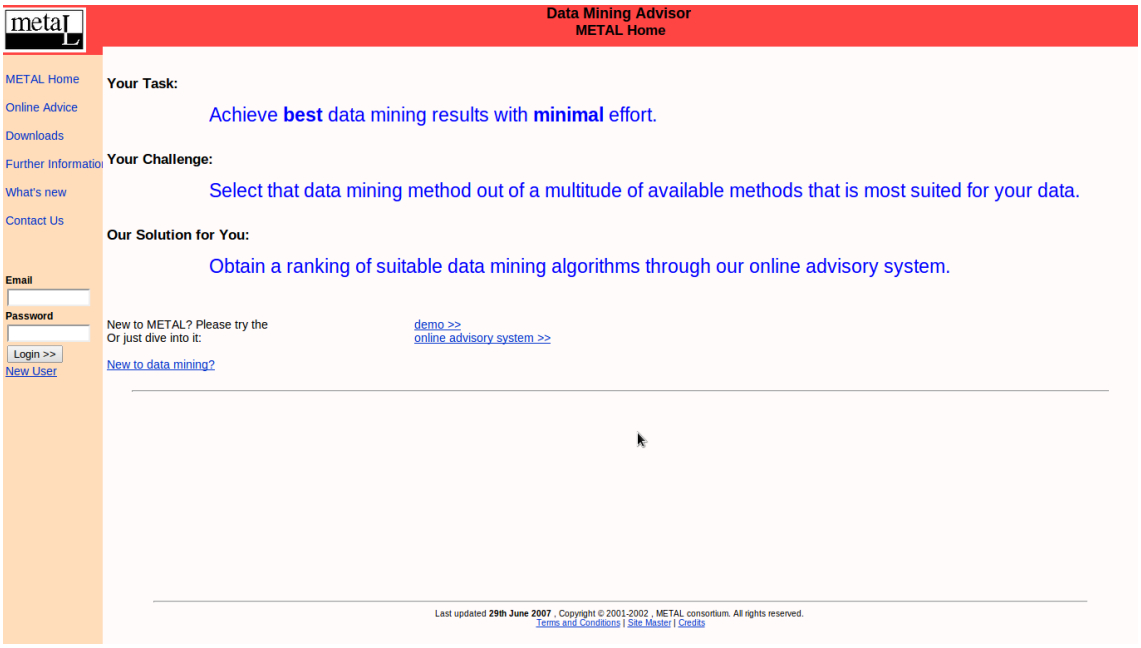

Figure 2.17: Data Mining Advisor

have been addressed many times in software reuse literature. However, the focus of these methodologies is directed to the code level rather than the underlying domain concepts.

2.3.4 Frameworks and Paradigms

We will now explain the concept of frameworks, and give a brief presentation of the chosen technology to develop this dissertation.

2.3.4.1 Frameworks

Frameworks are a universal, reusable software platform to develop applications, products and solutions. They supply generic functionality for the creation of software systems by providing a series of loosely-coupled components who can be overridden by user-written code, to allow application-specific software.

Usually, frameworks include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution.

Frameworks improve the quality, reliability and robustness of new software. They also improve the developer productivity by allowing them to focus on the unique requirements of their application instead of spending time on application infrastructure [WN13].

RoR [HF05, BK07] is a good example of a popular framework that aims to cut development time and costs in very different scenarios.

2.3.4.2 Ruby and Ruby On Rails

Ruby On Rails is a web development framework, but before we look into it, it might be pertinent to give a brief insight of its base paradigm Ruby.

2.3.4.2.1 Ruby Ruby is a dynamic, reflective, object-oriented programming language designed and developed by Yukihiro Matsumoto. Released to public in 1995, the author describes it as *"(...) a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write"* [Mat95].

Ruby syntax was inspired by many different programming languages, such as Eiffel, Lisp, C++, Perl and Python, to name a few. It possesses a series of characteristics that make it extremely attractive like dynamic type system and automatic memory management. Ruby also supports multiple paradigms including functional, object oriented, and imperative programming.

Designed to minimize confusion, its meta-architecture allows changes to class definitions in runtime, therefore constantly adapting by providing open, active class definitions. The language itself can be extended by writing higher level abstractions, which extend the core classes, for example a `String` or a `Float` with user custom methods [MO07].

The usage of meta-programming (section 2.3.3) techniques in Ruby, allows accessors methods to be created with a single line of code. Keeping all of its instance variables completely private to the class, Ruby only exposes such instances through the accessors methods, thus avoiding the need of `getters` and `setters`.

2.3.4.2.2 Ruby on Rails Ruby on Rails is an open source web framework "*(...) that's optimized for programmers happiness and sustainable productivity. It lets you write beautiful code by favouring convention over configuration*" [Han03], which runs on the Ruby programming language. Developed by David Heinemeier Hansson in 2003, Ruby on Rails, often simply Rails, is based on the model-view-controller (MVC) design pattern.

Chapter 3

DM4D

As explained earlier, the DM Advisor (Section 2.2.2.6) is a system for algorithm recommendation with some interesting features and concepts, but some characteristics need improvement.

In this project a system was developed to address two of those shortcomings. The first is to take advantage of the large amount of meta-data contained in OpenML (Section 2.2.2.1), the second is the scalability of the system by enabling the distributed execution of algorithms on resources that are provided by the users. In addition to these issues, a default algorithm server was also developed to fully implement the distributed algorithm execution strategy.

Developed in Ruby on Rails (Section 2.3.4.2) with a MySQL ([Man10], [Tiz08], [Jam01]) database, DM4D was designed to be easily scalable for developers, while easy to use and intuitive for the end-users. As seen in section 2.3.4.2.1, ruby is an elegant paradigm that tries to minimize confusion [MO07]. When combined with rails (Section 2.3.4.2.2) with its MVC design pattern, the platform is divided into three interconnected parts, the model, the view and the controller. This facilitates scalability, because it gives the platform a more modular design by subdividing the system into smaller parts. Thus, making it easier for adding and removing components. The platform was also projected to take fully advantage of a distributed execution strategy. To facilitate the communication for this strategy, an API was developed.

In the next sections a detailed explanation of the platform architecture and its implementation will be presented . In the last section we will present the developed algorithm server. A representation of the system architecture is presented in figure 3.1.

3.1 Architecture

As discussed before, OpenML (2.2.2.1) is a very interesting source of knowledge with data from more than 575.000 experiments. To take advantage of such knowledge, and to respond to the issues identified in Section 1.1, we collected significant meta-features from the OpenML database and built our own meta-database (MetaDB). By applying a meta-learning (Section 2.1.2) technique,

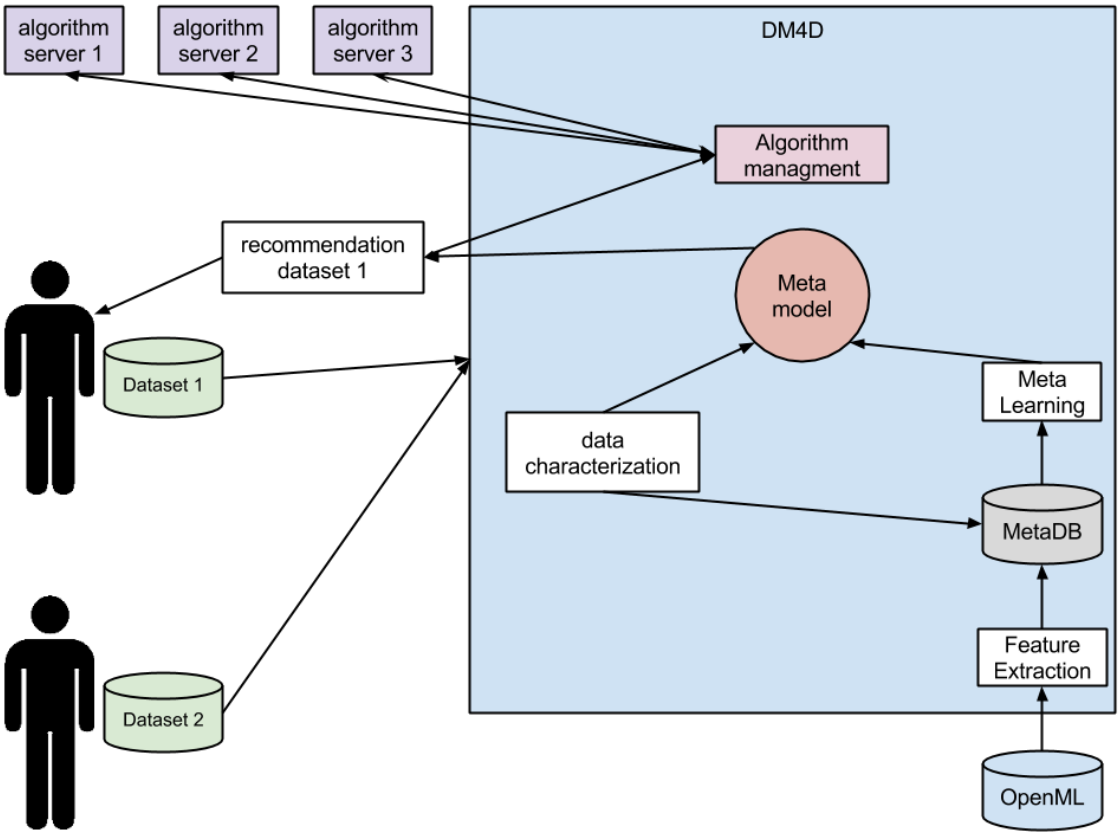

Figure 3.1: Architecture of the proposed approach.

a model was built from the created MetaDB. This model is not only able of learning from past experiments but will also be able to learn from the future experiments that will be made. This learning ability is possible because the model is directly connected with the MetaDB whose data is updated throughout each experiment, or when synchronized with OpenML. Such adaptability will help our model to provide better recommendations throughout the time.

To facilitate this learning ability, a web base platform was developed where the model is integrated. Upon receiving a new dataset, it will extract its characteristics and invoke the embedded model, mapping the dataset to the most promising algorithms. The recommendations are then presented in a ranked list (from the most promising to the least).

Finally, the user is then able to request the execution of a specific algorithm on the dataset, if the algorithm is available. Functioning has a broker, whose concept was inherited from the DM Advisor (Section 2.2.2.6), the system will then request to the correspondent algorithm server to execute the algorithm on the dataset.

To complement the platform, and help avoiding knowledge loss between experiments, a *default algorithm server* was developed to aid the platform to execute each experiment in an automated way. This default algorithm server is capable of receiving execution requests, execute the algorithm on a specific dataset and communicate the experiment result back to the platform.

```
1       SELECT r.rid, d.name FROM run r, input_data rd, dataset d WHERE d.did = rd.
            data AND rd.run = r.rid
```

Listing 3.1: SQL query to retrieve run records

3.1.1 Database

Has mentioned in section 3.1, DM4D has a MySQL database. MySQL is the world's most popular open source database, and second most used[1].

The database structure was inherited from DM Advisor (2.2.2.6). The schema of the database is presented in figure 3.2. However, the original structure presented various limitations and constrains, like traceability, scalability, management and synchronization. Thus, the necessary improvements were made. Some of the most important changes are the addition of `created_at` and `updated_at` fields to every table facilitating traceability, the possibility to assign a user to multiple roles thus allowing a better management and the addition of `openml_id` field in the necessary tables for synchronization with OpenML (Section 2.2.2.1). Many other changes and additions were made, like `invitation_token` and `reset_password_token` fields, but these will not be specified due to its low relevance to the main architecture.

To feed our MetaDB, firstly we had to identify which data, contained in OpenML (Section 2.2.2.1), was significant for DM4D. However, during this dissertation OpenML was in the process of making some major changes to its current schema. This was identified as a risk who could take the development a few steps back if the migration occurred during the development. In order to bypass this risk, we retrieved an SQL dump from OpenML and performed our work with the collected image.

Analysing the OpenML schema (Figure 2.6) and experimenting with the OpenML image, we identified that every time an algorithm is executed on a dataset, a record is created in the table *run*. This record is linked to several records of the tables *dataset* and *evaluation*. The *input_data* and *output_data* are the linking tables that define a many-to-many relationship between those tables. The performed query can be seen in Listing 3.1. However, most classification tasks contain just one input, so OpenML created a materialized view, the table *cvrun*. The table contains for every run a record, including a link to the dataset, enabling faster queries. The equivalent query to retrieve the same data as in Listing 3.1 is presented in Listing 3.2. Next, we wanted to include the algorithm used for the run. This is recorded in the *algorithm_setup* table. It contains all information about the algorithm that was run. More precisely, what implementation was used, and what the parameter settings were. Extending the query 3.2 to 3.3 it was possible to match the learning algorithm that was used through the *learner* column. Finally, the performance measures of the datasets were retrieved. The used query can be seen in Listing 3.4.

[1] Following SQLlite, which is deployed with every iPhone and Android device and with the Chrome and Firefox browsers.

```
1       SELECT r.rid, d.name FROM cvrun r, dataset d WHERE r.inputData = d.did
```

Listing 3.2: CVrun query

3.1.2 Meta-Learning

An important part of the platform is its capability to extract dataset characteristics and map them to the most promising algorithms. A representation of the process can be seen in figure 3.3.

Before being able to *predict*, the model needs to *learn*. The model retrieves the performance of algorithms on datasets, the model *learns* what *types* of datasets the algorithm best performs on. This is done utilizing the meta-learning algorithm kNN seen in section 2.1.1. When a new dataset is received, the model receives the extracted meta-features from DM4D of the new dataset. Afterwards, the model retrieves all the existent datasets and its characterization from the DM4D MetaDB. Applying the kNN (Section 2.1.1) algorithm, the new dataset is assigned to a group of known datasets whose characteristics are similar to its own. The model then retrieves data about the performance of all the algorithms, for each of the datasets present in the selected group, and predicts a ranking of which algorithms will probably best perform on the newly uploaded dataset. Finally, the ranking is stored in the MetaDB and presented to the user.

The model was developed in R (Section 2.2.1.3), and is also directly connected to the platform database, giving it awareness of newly collected data. Such information is then used to learn more about an algorithm performance on the different datasets, allowing our model to keep learning and give better predictions.

3.1.3 Communication

Before allowing the user to request an execution, the platform starts by checking if the algorithm server is reachable. If so, DM4D creates a new *run*, and stores it on the database. After, it generates a JSON ([Cro06], [CZ10], [BZ12]) message and sends it to the algorithm server. Each request, or JSON object, contains the information about the *run*. An ID field that identifies the run in DM4D, the URL where the dataset is located and the command the algorithm server should execute. An example of request message from the platform to an algorithm server is:

```
{id:<run id>, url:<dataset url>, command:<execution_command>}
```

```
1       SELECT r.rid, d.name, s.implementation, e.value FROM cvrun r, dataset d,
            output_data re, evaluation e, algorithm_setup s WHERE r.inputData = d.did
             AND r.rid = re.run AND re.data = e.did AND e.function = "
            predictive_accuracy" AND r.learner = s.sid
```

Listing 3.3: CVrun query with algorithms

```
1       SELECT r.rid, d.name, e.function,e.value FROM cvrun r, dataset d, output_data
             re, evaluation e WHERE r.inputData = d.did AND r.rid = re.run AND re.
            data = e.did
```

Listing 3.4: Performance measures query

After sending the request, the platform informs the user that the request as been made and waits for a response. By delegating the responsibility of replying back the result to the algorithm servers, it frees the platform from excessive and unnecessary work of controlling each execution. The results can be easily communicated through the DM4D API. We will discuss it in Section 3.1.4. A representation of the communication flow between the platform and a algorithm server can be seen in figure 3.4.

3.1.4 API

An application programming interface (API), is typically a set of HTTP request messages, with a definition of the response structure, usually in a XML or JSON format.

DM4D API was developed under the REST architectural style (Section 2.3.1.2). It allows users the possibility to develop their own automated processes. The API calls are performed as if accessing the platform from a common browser but terminating the URL with the response format. For instance, if a user was navigating through the datasets, the URL would be something like:

`http://dm4d.fe.up.pt/datasets?page=2`

This call would return a set of HTML files with the retrieved content to be interpreted by the browser as in figure 3.5. The same call to the API would be:

`http://dm4d.fe.up.pt/datasets.json?page=2`

However instead of returning a set of HTML files, the call returns an array of JSON objects containing the same information (figure 3.6).

3.1.4.1 Remote Procedure Calls

DM4D API as available a considerable set of operations. These, come in the form of RPC who can be initiated by sending a request message to the server to be executed with the supplied parameters. The most relevant are:

3.1.4.1.1 Dataset uploading To upload a dataset through the API, the user must send the request message to `http://dm4d.fe.up.pt/datasets.json` with the request method *POST*. In addition, the name, problem, identifier and dataset parameters must also be supplied. The server will reply with the specified response format. If the call is successful, the returned object will contain the id assigned to the uploaded dataset. If not, and error object will be returned.

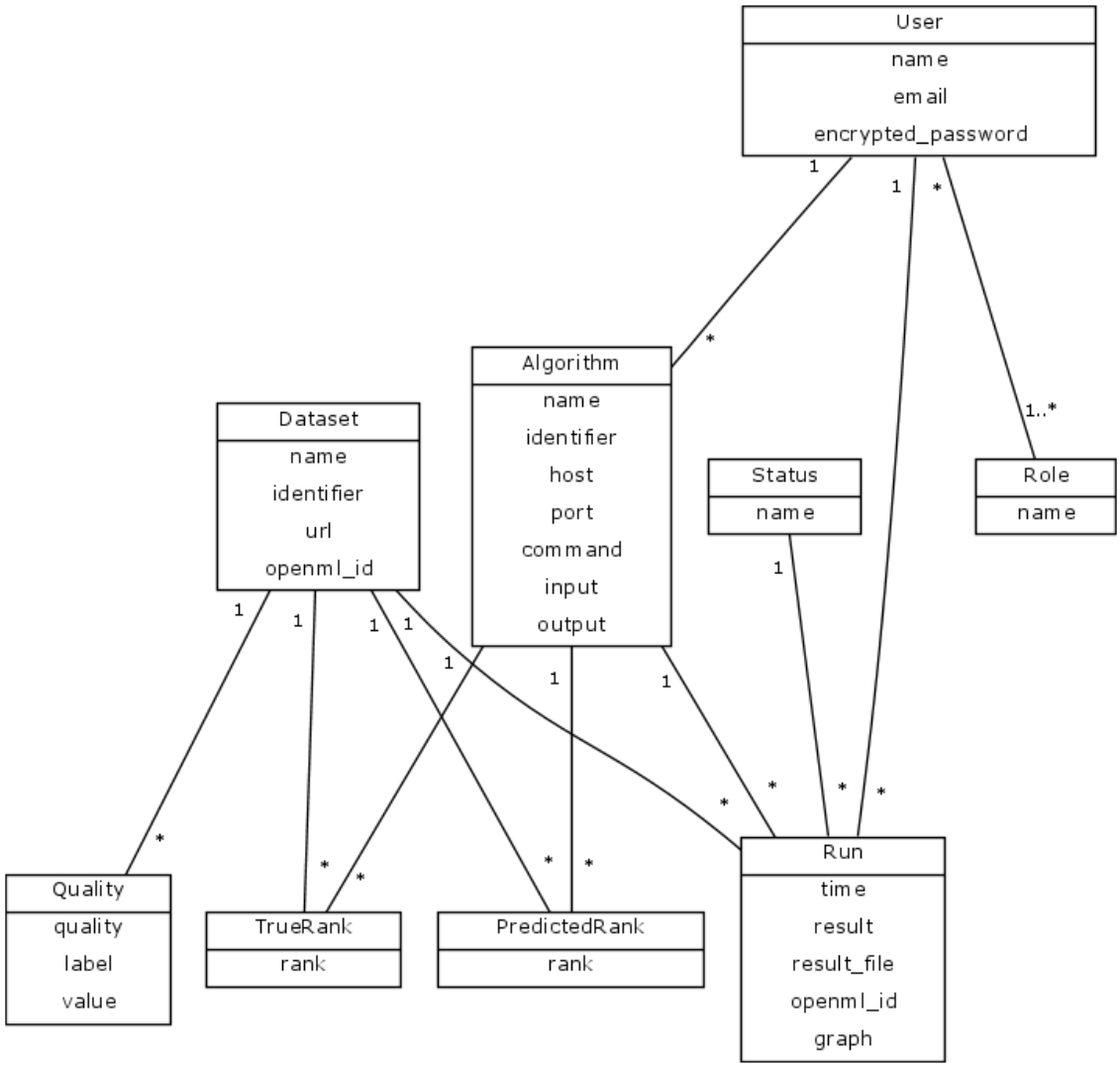

Figure 3.2: Architecture of the DM4D database (MetaDB).

3.1.4.1.2 Recommendation To retrieve the recommendations of a dataset, a request message must be sent to `http://dm4d.fe.up.pt/datasets/id.json` with the request method *GET*. The only parameter that must be supplied is the dataset `id`. Replying with the specified request format, the server will reply an object containing information about the dataset and a list of algorithms ordered by the predicted value (rank).

3.1.4.1.3 Request run To request the execution of a algorithm on a dataset, the request message must be done to `http://dm4d.fe.up.pt/run` with the request method *POST*. In addition, the algorithm and dataset parameters must also be supplied. If the request was successful the server will reply with the `id` of the run in the specified response format. If not, an error message will be returned.

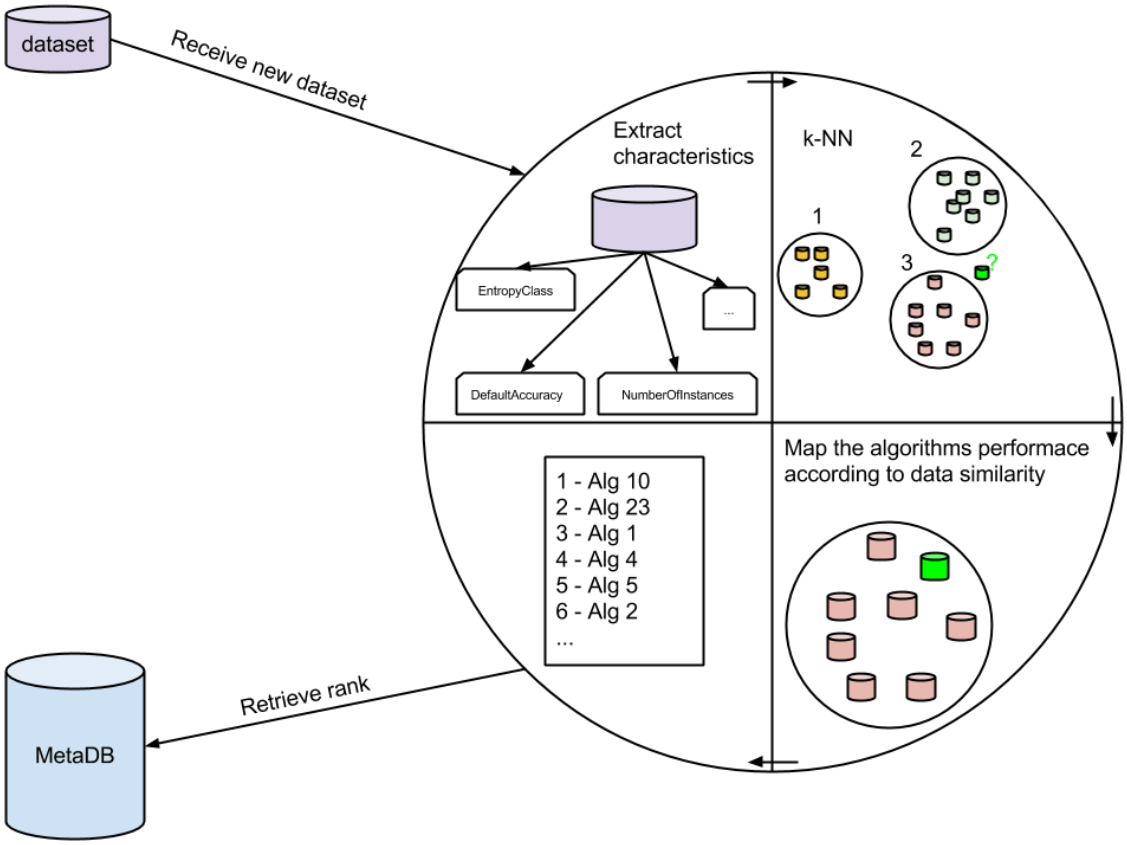

Figure 3.3: Meta-Model work-flow.

3.1.4.1.4 Retrieve run result It is possible to retrieve information about a run. Sending a request message to `http://dm4d.fe.up.pt/runs/id.json` with the request method *GET*, the server will reply with the specified request format. The information contains the current known run state and the run result, among others.

3.1.4.1.5 Result Uploading To update the result of a run the request message must be sent to `http://dm4d.fe.up.pt/runs/id.json` with the request method *PATCH* or *PUT*. The id the run and the result parameters must also be supplied. The response message will contain if the result submission was successful or not.

3.2 Default Algorithm Server

As discussed earlier, DM4D tries to address the scalability issue present in DM Advisor (section 2.2.2.6), while providing the integration of new algorithms and availability of algorithm execution functionalities. To address this issue the idea of algorithm servers spurred. This, lead to the API (section 3.1.4) for a way to simplify the integration of existing algorithms, which in its turn, lead to

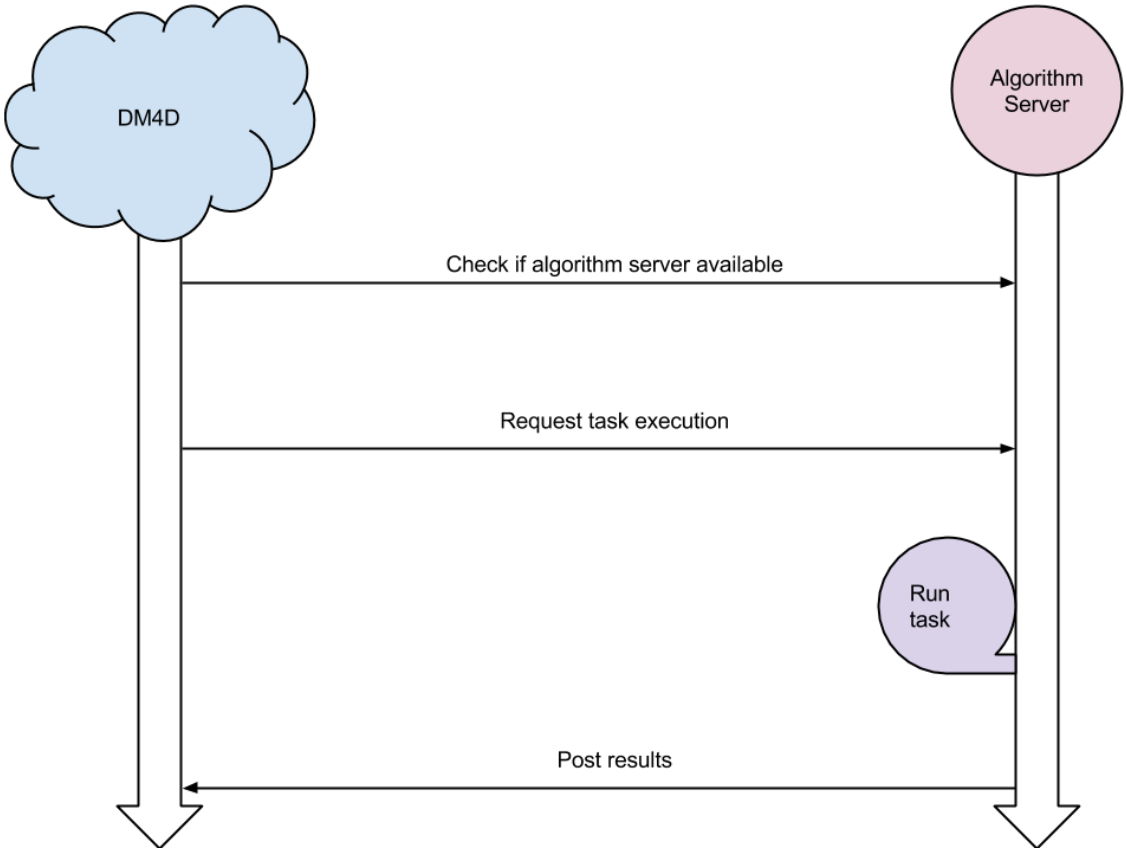

Figure 3.4: Communication flow between DM4D and a algorithm server.

the idea of remote execution and collection of results. To reply to this necessity, the development of a default algorithm server (DAS), which is a wrapper for existing algorithms, was performed.

Developed to be capable of being executed in any machine that supports *Java* ([GJSB05]), making it extremely system wide compatible, DAS acts has an interpreter between the web-base platform and the algorithm it is responsible for. It gives the user the choice of adding its own algorithms without compromising the source code. This way, a user can compare its algorithm performance against others algorithms in various datasets, while sharing its results, and make the algorithm available to the community through a platform which contains many more algorithms.

DAS was designed to be fault tolerant, and capable of executing each requested task asynchronously. Upon receiving a new request the default algorithm server adds the information received from the server to a queue. When resources become available, the default algorithm server will start the execution in a thread[2]. The thread is responsible for downloading the dataset, execute the task, retrieve the result and communicate it back to platform through the API (Section 3.1.4). Executing each task independently from DAS, reduces the risk of DAS being compromised by a faulty execution, a failed download or any other factor. Thus, being able to process its current, or

[2]For more information about Java threads please see [Mil09], [GP06], [WPM+09] and [TSD09]

Figure 3.5: Browser navigation example.

new, queued tasks and try to execute each one whenever possible. To add a new task to the default algorithm server, the request has to be sent in the format seen in section 3.1.3.

The default algorithm server, upon receiving the request adds it to his queue. When possible the default algorithm server executes the request task, and post the execution result back to the platform.

A figural representation of the default algorithm server architecture is presented in figure 3.7.

To execute the default algorithm server it is necessary that the algorithm itself respects some directives:

- The binary executable file of the algorithm has to be present in the specified folder received from the request. This is done so that a user can supply multiple algorithms from the same machine.
- The algorithm has to store the execution results in a new file with the following name format *datasetName.ARFF.result*, ex: `Abalone.ARFF.result`. Inside the file, the execution result has to be stored with the following format: `result:0.0667050355114043`.

3.2.1 Customization

DAS is powered with a few customizable configuration options:

- `-p` The port to use to communicate with the DM4D web-based platform (ex: `-p 12345`). This allows the possibility for the user to have more than one active DAS communicating through different ports. However, DAS can also manage multiple algorithms throughout the same port. This provides the users with a more diverse management possibility. For example, a user can have on the same machine two DAS servers running and each DAS is responsible for two different algorithms, which in total four different algorithms under the same machine.
- `-f` The folder where the datasets should be downloaded to. By default DAS downloads the necessary datasets to the same location he was executed. However these datasets can occupy a large amount of space, so it is advised specifying the download folder for better management. Ex: `-f /tmp/`

```
[
  - {
        id: 31,
        name: "credit-g",
        problem_id: 1,
        identifier: null,
        url: "http://localhost:3000/datasets/31.json",
        openml_id: 31
    },
  - {
        id: 32,
        name: "pendigits",
        problem_id: 1,
        identifier: null,
        url: "http://localhost:3000/datasets/32.json",
        openml_id: 32
    },
```


Figure 3.6: API response example.

- `-m` The number of runs the default algorithm server can start at the same time. Nowadays computers are ever more capable of executing multiples tasks at the same time. Ex: `-m 3`
- `-u` The host-name. By default DAS works with the DM4D API. However, for testing purposes, execution experiments or any other possible reason, the user can specify the host-name and the RPC he wishes DAS to communicate with.

  Ex: `-u www.dm4d.com/new_result.json`
- `-h` The default algorithm server can also receive a command that will just show the available options upon start.

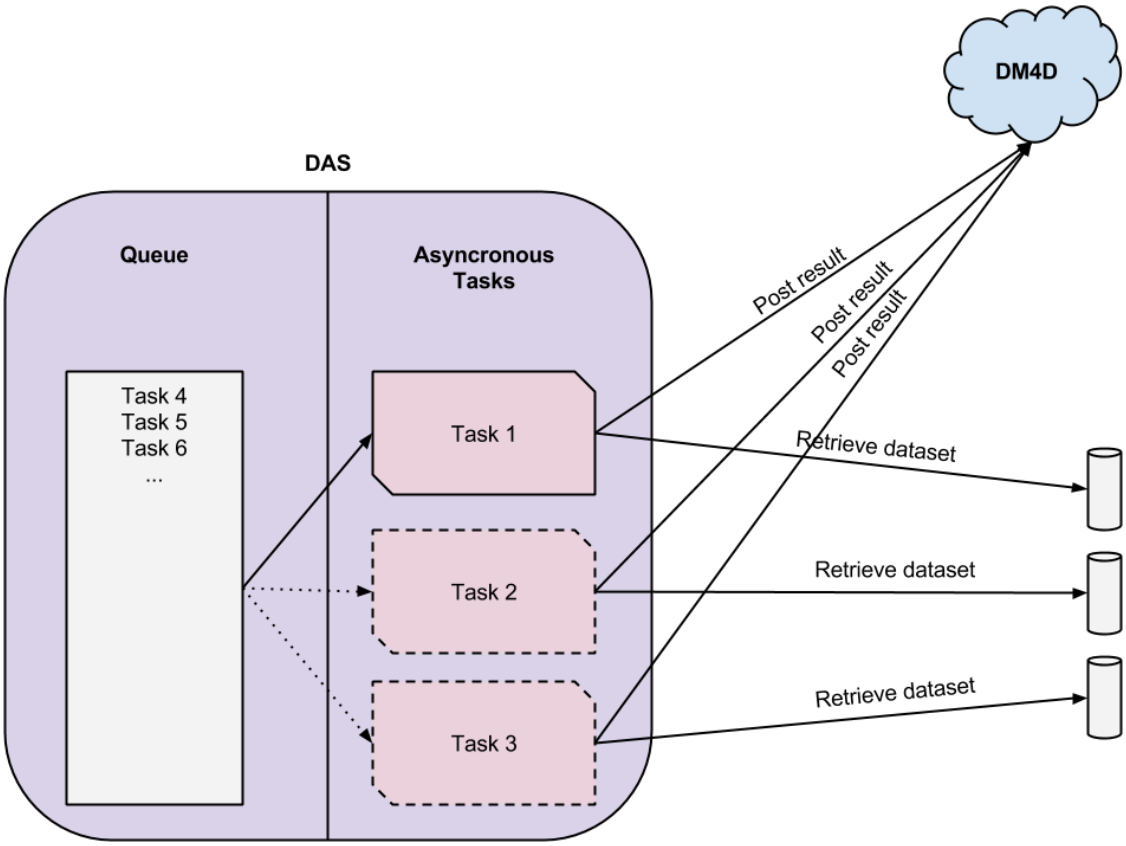

Figure 3.7: Default algorithm server architecture.

Chapter 4

Use Cases

Understanding the advantages of using different data mining tools and techniques is an important task for every *miner*. This becomes even more crucial when the focus of the tool is guiding novice data miners in the right direction. Additionally, is the need that the user has already overcome the learning step of how to install, configure and use such tools.

This need of background knowledge may truly overwhelm the novice *miner*. However, we can reduce this feeling by allowing the user to experiment in a secure and familiar environment, like a browser.

In this chapter we will explain some of the most common use cases. For each use case we will present its guidelines and expected results. In the end we will explain how we validate the obtained results.

4.1 Registration and Authentication

Although the platform allows obtaining a recommendation on an existing dataset and navigating through the data without being authenticated. It is necessary for all other tasks to be registered and authenticated. This is a necessary constrain, because it facilitates management and administration.

The registration is a simple process. To register, the user simply to press the *Sign up* button (figure 4.1). This will redirect the user to the registration form where he will have to supply his name, email and password (Figure 4.2). It is necessary that the email address be a valid one. Before being able to authenticate in the platform, a confirmation link will be sent to the email. Only after confirming this the account will the user be able to authenticate.

4.2 Authentication

The authentication process is a very simple and common process process. The user simply has to go to the *Login* button located in the top-bar (figure 4.1). This will redirect the user to the

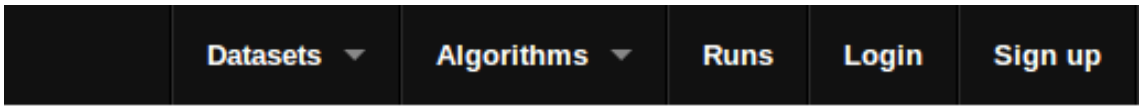

Figure 4.1: *Login* and *Sign up* buttons of the top-bar.

authentication form (figure 4.3) where he will simply have to supply his registered email address and chosen password.

4.3 Upload a Dataset

When in possession of a dataset, a user needs to choose which algorithm(s) he should, or can, use. To be able to get recommendations the user must first upload the dataset to the DM4D platform. To do this a user has two options, either from the *home* page selecting the *try me* option (figure 4.4). Alternatively, from anywhere within the platform, by going to the top bar, choosing *Datasets* (option 1) and *Datasets* (option 2) as showed in figure 4.5). Either way, this will take the user to the datasets listing (figure 4.6). From there the user has to select the *New Dataset* option located on the left of the list (option 1 in figure 4.6). The *new dataset* form (figure 4.7) will then be presented, where the user can provide some details and upload his dataset.

4.4 Algorithm Recommendation

After uploading the dataset to DM4D, the platform will generate its predictions. If it is a newly uploaded dataset, the platform will automatically redirect the user to the its recommendations page (figure 4.8). However, a user may want to get recommendations from already uploaded datasets. In this case, the user simply needs to follow the steps from figure 4.5 who will take him to a listing page with all the datasets present in DM4D. From there, he simply has to navigate through the page (figure 3.5) to locate the desired dataset and select it. This will take the user to the dataset page where he will simply have to select the *rank* tab (figure 4.8) to be presented with a rank of algorithms.

4.4.1 Assessing Ranking Accuracy

To measure the predicted rank and the true rank relation, the platform uses the Spearman's rank correlation coefficient (section 2.1.2.1). The coefficient is calculated for each dataset, giving a direct clear view of the correlation between the predicted and true ranks. The closer to $+1$ the closer the predictions were to their true rank. If the coefficient is more closer to the -1 then the predicted rankings are in an inverse order. If the coefficient is 0 than the rankings have no relation

Figure 4.2: Registration form in DM4D.

at all and the ranks are completely different. However, this measure can only be computed if the results are available in OpenML or if the algorithms are executed in DM4D.

4.5 Dataset Exploration

A user may want to explore a dataset. Check the its meta-data, algorithm performance on the dataset, who requested the run, among others. After following the steps in figures 4.5 and choose a dataset from the dataset list (figure 3.5), the user will have 3 panel he can choose from. By default the second panel will be selected were the algorithm recommendations are presented. If he selects the first panel, he will be able to explore the datasets meta-data like in figure 4.9. In the third panel its presented a list of runs. These are the runs that have been executed on the dataset along with some extra details (figure 4.10).

4.6 Requesting a Run

Given the recommended ranking of the algorithms for a dataset (figure 4.8), the user may want to execute some algorithms. As we can see in figure 4.11 the platform provides very simple and straightforward guidance. The user simply has to press a *request run* button and the platform will do the rest. The user will receive a notice informing that the request has been made (figure 4.12). Additionally, the user will also know wich algorithms have been request to be ran on the dataset by having the *Running* text in the algorithm row (figure 4.13).

4.7 Add New Algorithm

The platform also provides interesting opportunities for developers of learning algorithms. For example, because the platform ranks the algorithms according to their performance on each dataset, a *miner* may want to test its own algorithms against the ones that are present in the platform.

Figure 4.3: Authentication form in DM4D.

To add a new algorithm to the platform, a user needs to be registered and authenticated. This is a necessary step for it is important that the system knows who is responsible for which algorithm. Just like in uploading a new dataset (4.3), the user can access from anywhere within the platform, by going to the top bar, choosing *Algorithms* (option 1) and *Algorithms* (option 2) as in figure 4.14. This will take the user to the algorithms listing (figure 4.15). From there the user has to select the *New Algorithm* option located on the left of the list (option 1 in figure 4.15). The *new algorithm* form (figure 4.16) will then be presented, where the user can add his algorithm.

It is important to refer that the fields `Host`, `Port` and `Command` are extremely important in order for the algorithm server to be able to execute the algorithm and retrieve back the result. The `Host` is the IP[1] of the algorithm server. The `Port` is the communication endpoint from where our host is *listening*. The `Command` is the instruction the algorithm server will execute for each request (example `Rscript test.R`).

4.8 Executing DAS

After, or before, adding the algorithm to the platform, the user needs to start the default algorithm server. As seen before (section 3.2), DAS comes powered with a few configuration options, but for this use case we will explain the default behaviour. As also explained in section 3.2, DAS runs under the Java Virtual Machine (JVM), so before trying to start the default algorithm server, the user needs to install the specific JVM of his Operating System (OS). After successfully install the JVM, the user simply has to execute the following command `java -jar AlgExec.jar`, followed by the desired options (section 3.2.1).

[1] Its the computer address under the Internet Protocol.

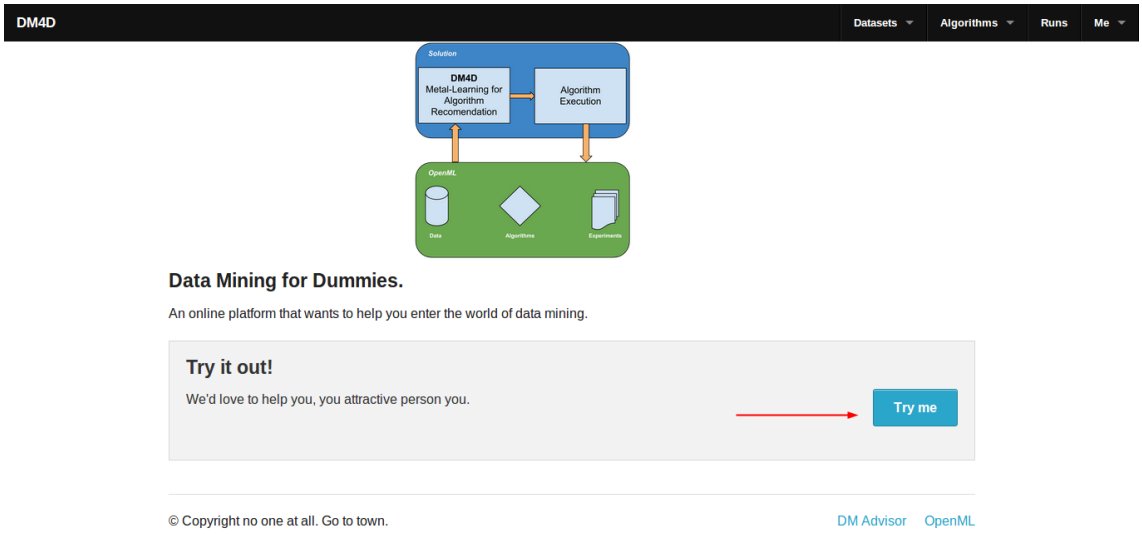

Figure 4.4: Selecting the *Try me* option

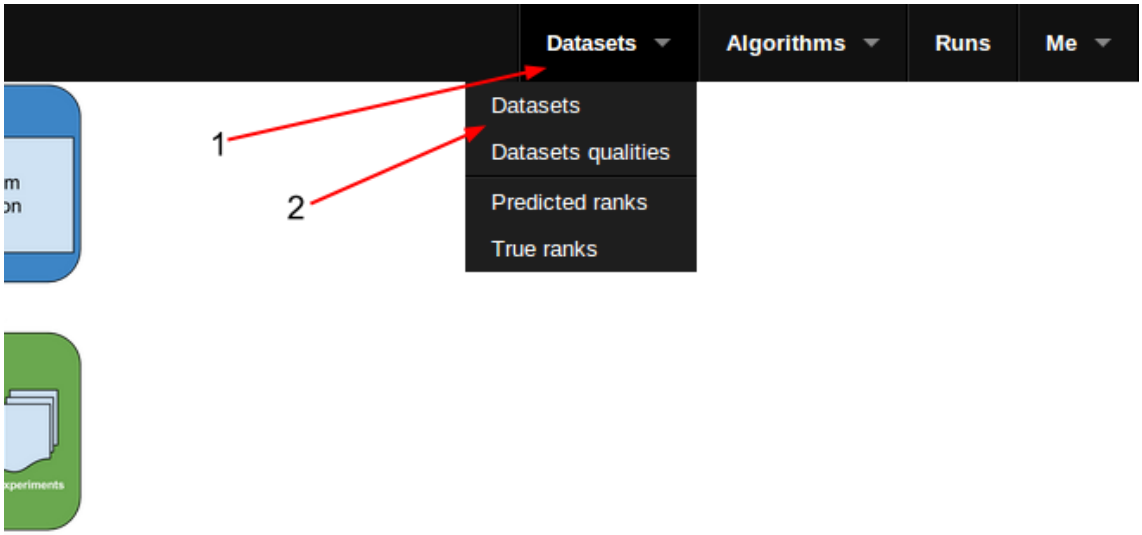

Figure 4.5: Selecting the *Datasets* option

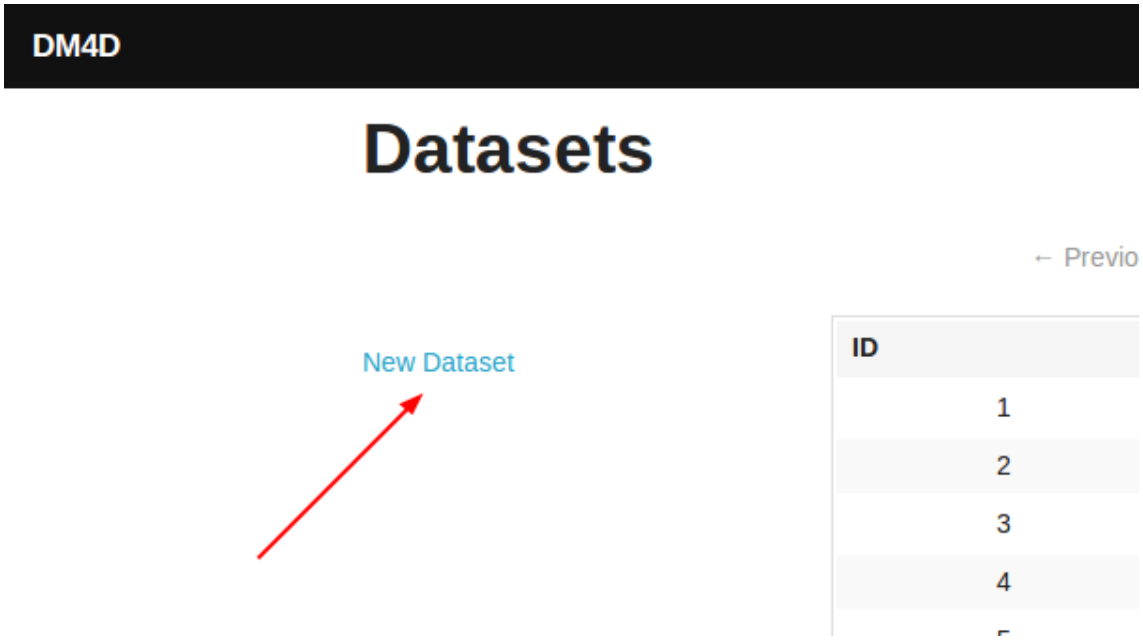

Figure 4.6: Selecting the *New Dataset*

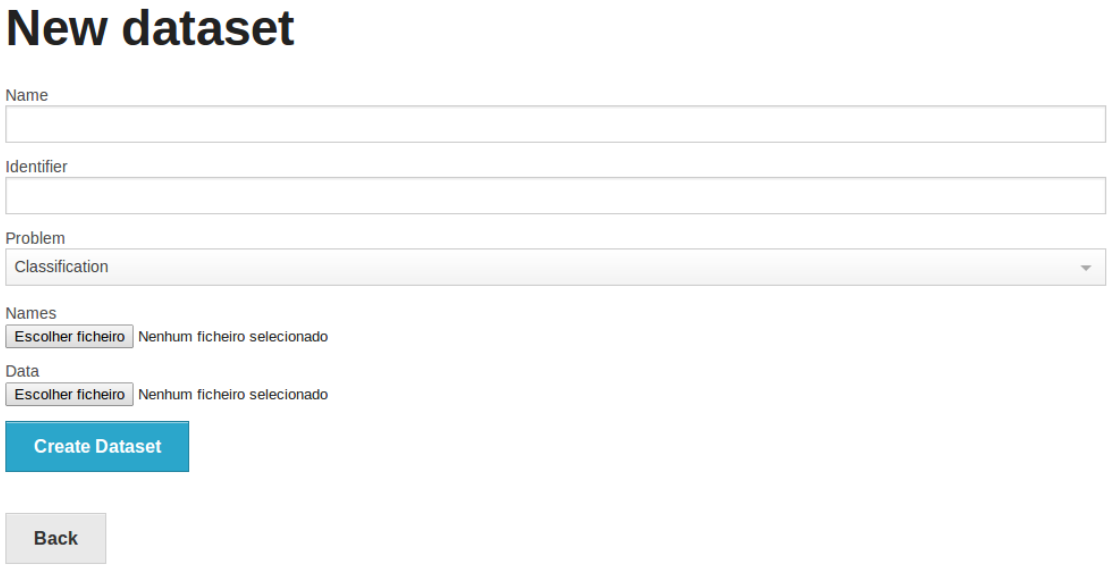

Figure 4.7: Selecting *New Dataset*

Qualities | Rank | Runs

Algorithm	Predicted Rank	True Rank
weka.Bagging(1.31.2.2)	1	
weka.DecisionStump(1.18)	2	
weka.ConjunctiveRule(1.10)	3	
weka.HyperPipes(1.15)	4	
weka.LogitBoost(1.33)	5	
weka.AdaBoostM1(1.24.2.3)	6	
weka.AttributeSelectedClassifier(1.16.2.4)	7	
weka.CVParameterSelection(1.27.2.3)	8	
weka.IB1(1.13.2.1)	9	
weka.OneR(1.17)	10	
weka.IBk(1.32)	11	
weka.FilteredClassifier(1.20.2.2)	12	
weka.MultiScheme(1.18.2.1)	13	
weka.J48(1.2)	14	

Figure 4.8: Algorithm recommendations rank

Qualities | Rank | Runs

Quality	Label	Value
DefaultAccuracy		0.04065
EntropyClass		4.69981
FeatureAbsoluteSkewness	max	1.12094
FeatureAbsoluteSkewness	min	0.0106007
FeatureEntropy	max	4.69981
FeatureEntropy	mean	4.69981
FeatureEntropy	min	4.69981
FeatureKurtosis	max	6.66511
FeatureKurtosis	mean	0.392133
FeatureKurtosis	min	-0.738783
FeatureSkewness	max	0.991943
FeatureSkewness	mean	0.124853
FeatureSkewness	min	-1.12094
JointEntropy	max	0
LandMarker1NN		0.9588

Figure 4.9: Known meta-data of the dataset.

Qualities | Rank | Runs

← Previous 1 2 Next →

ID	User	Algorithm	Result	Result File	Problem	Status	Graph	Time
40480	First Admin	weka.DecisionTable(1.29.2.1)	0.71535		Classification	Finished		0
40481	First Admin	weka.NNge(1.2.2.1)	0.9143		Classification	Finished		0
40482	First Admin	weka.DecisionStump(1.18)	0.0709		Classification	Finished		0
40483	First Admin	weka.SMO(1.53.2.2)	0.8234		Classification	Finished		0
40484	First Admin	weka.MultilayerPerceptron(1.2)	0.82075		Classification	Finished		0
40485	First Admin	weka.MultiBoostAB(1.6.2.2)	0.0709		Classification	Finished		0
40486	First Admin	weka.ConjunctiveRule(1.10)	0.0703		Classification	Finished		0
40487	First Admin	weka.Stacking(1.23.2.1)	0.04065		Classification	Finished		0
40488	First Admin	weka.J48(1.2)	0.8798		Classification	Finished		0
40489	First Admin	weka.Decorate(1.3.2.1)	0.91475		Classification	Finished		0
40490	First Admin	weka.PART(1.2.2.1)	0.8876		Classification	Finished		0
40491	First Admin	weka.ZeroR(1.11)	0.04065		Classification	Finished		0
40492	First Admin	weka.BayesNet(1.21.2.4)	0.74365		Classification	Finished		0
40493	First Admin	weka.SimpleLogistic(1.5.2.1)	0.7741		Classification	Finished		0

Figure 4.10: Algorithm runs on the dataset

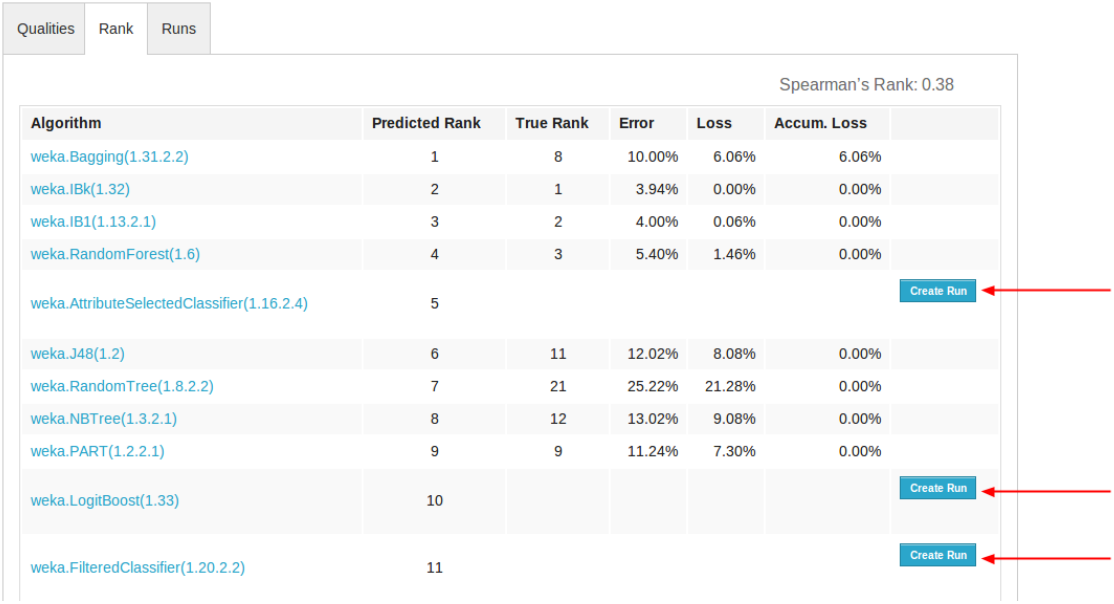

Figure 4.11: Choosing the algorithms to run on the dataset

Figure 4.12: Informing the user, the execution request was made

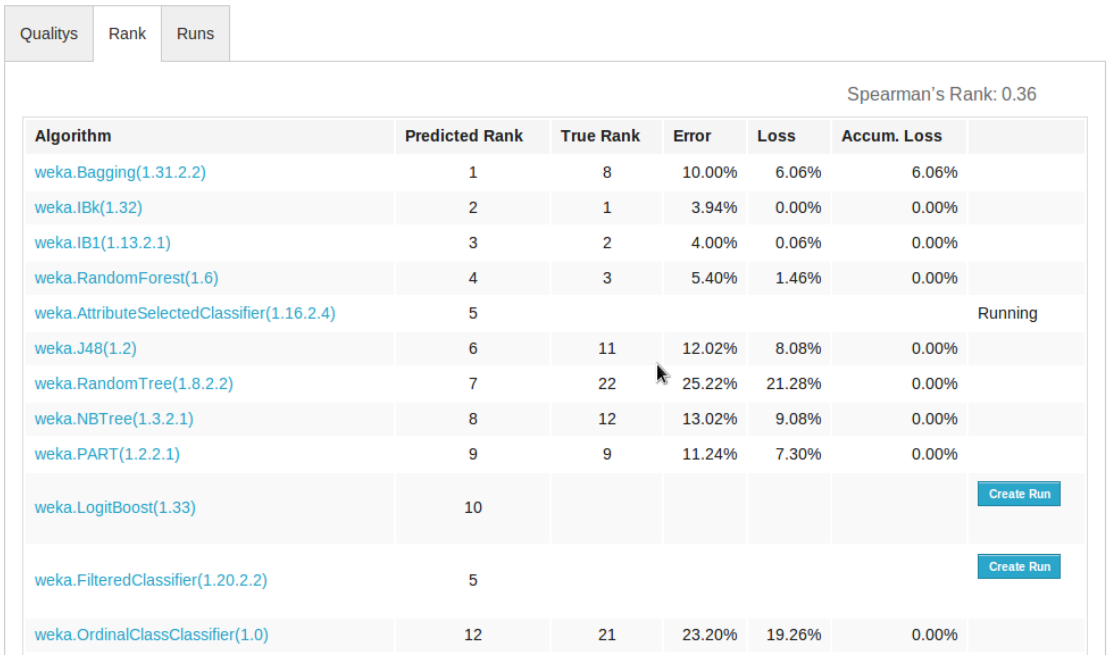

Figure 4.13: Visual information that an algorithm has been requested to run on the dataset.

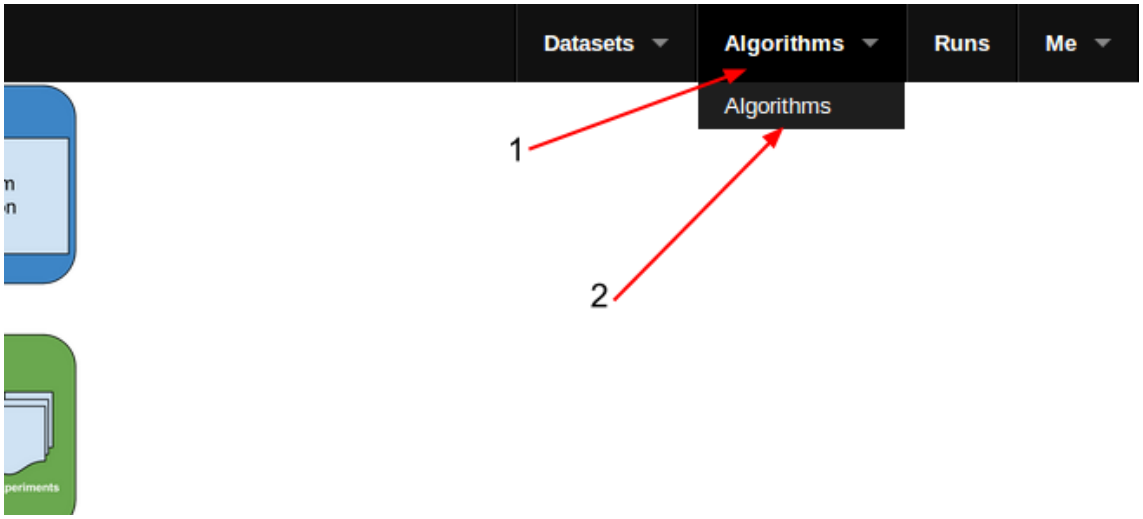

Figure 4.14: Selecting the *Algorithms* option

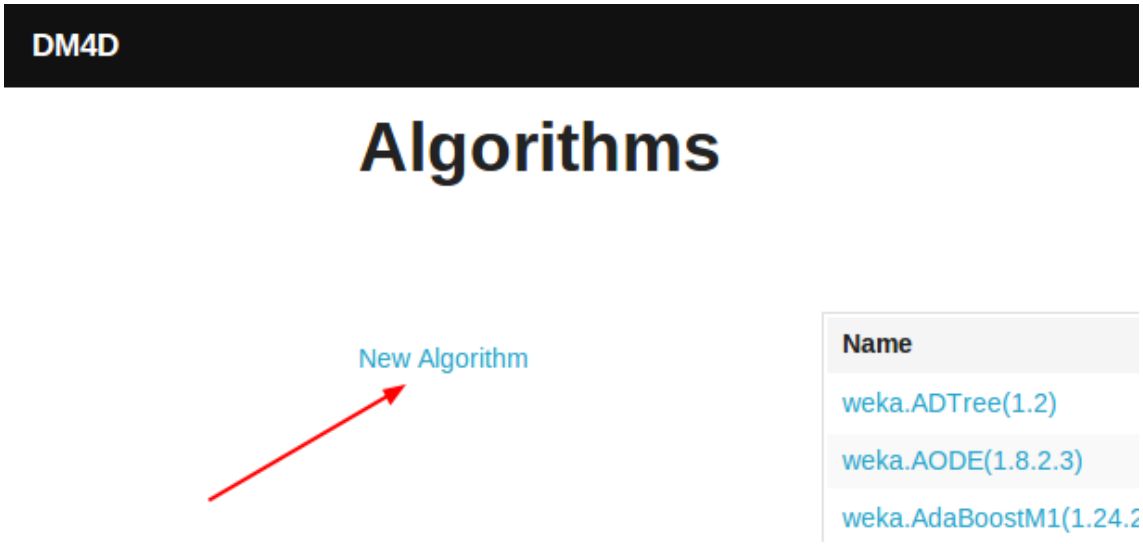

Figure 4.15: Selecting the *New algorithm* option

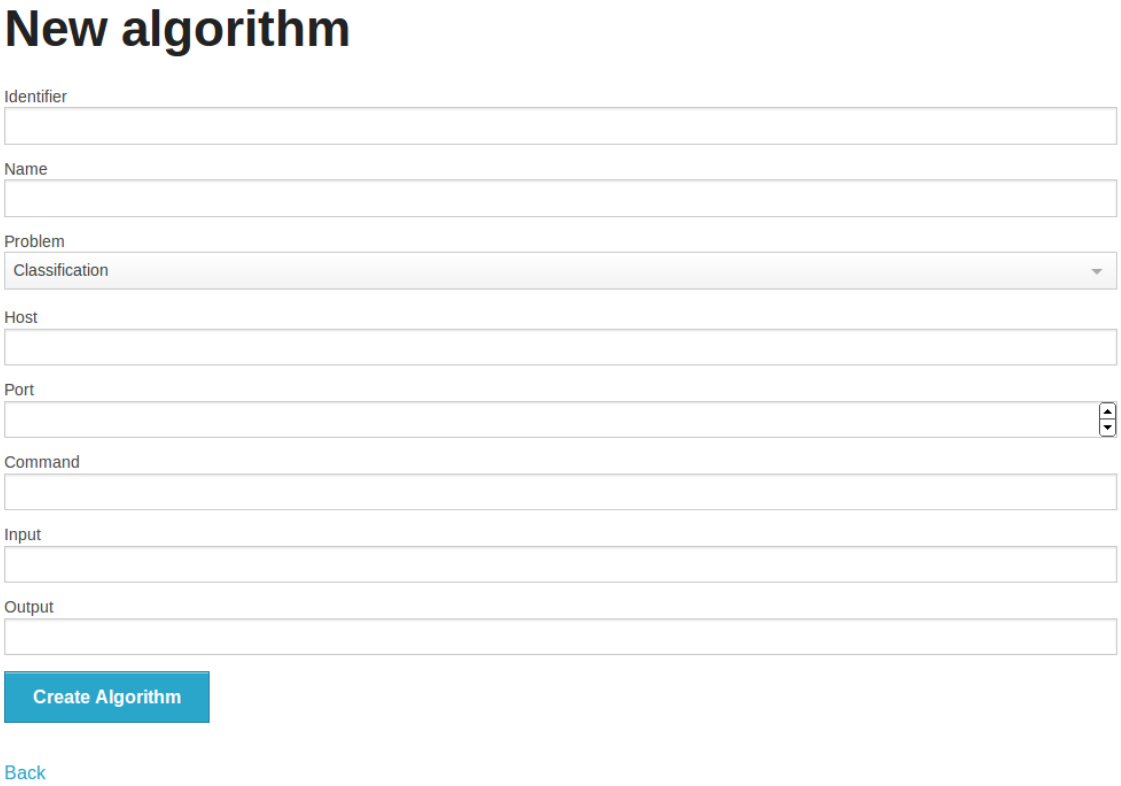

Figure 4.16: New algorithm form

Chapter 5

Conclusions

In this chapter it will be presented a summary of the project, and the accomplished objectives. Finally some final thoughts will be presented for future work.

5.1 Final Remarks

The goals of this project, as seen in section 1.2, was to improve and rebuild the DM Advisor (2.2.2.6), integrate it with OpenML (2.2.2.1), allow users to receive a recommendation for their dataset and execute some of the recommended algorithms on the uploaded dataset.

To accomplish these goals, this dissertation was developed in five different phases. The first was the development of a platform with a distributed architecture, which reduces the end-user from tool dependency and background knowledge. The second was the integration with OpenML and extracting its significant data. The third was the integration of the meta-model to learn from the platform and the OpenML experiments, powering the platform with the functionality of providing algorithm recommendations to new datasets. The fourth, was the development of an API to facilitate the communication with external tools. Finally, the development of DAS to facilitate the integration of new algorithms by providing a wrapper that can be used to call the algorithms and collect the results to send to the DM4D platform.

5.2 Future work

Although the dissertation objectives were completely met, there are some aspects that can be improved, whether by improving the meta-learning process or by extending the functionalities of the platform. A list of the most interesting issues to be done follows:

- **Data characterization** The meta-features extracted from each dataset, are the most common. If we added the extraction of more meta-features, like the kurtosis of each attribute, the mean kurtosis of attributes or the mean absolute skewness of attributes, this could allow

our model to have a better description of each dataset. If these meta-features carry useful information it might be possible that our meta-model would improve its understanding between datasets similarity, possibly reflecting in better predictions.

- **Recommendation learning** An interesting approach would be the that the model be able to learn about algorithms similarities and discover *rivalries* between them. In other words, if a algorithm `A` is good in `Dataset 1` and `B` is not, then on datasets were `A` does not have a good performance `B` will probably have. After a considerable amount of executions if a predicted algorithm `A` was ranked as 1 but his true rank was 12, it might be interesting to recommend algorithm `B` who is a *rival* of `A`.

- **Meta-learning techniques** The meta-learning technique used in the project is the k-NN algorithm. As seen earlier in section 2.1.1, this is one of the simplest algorithms in machine learning. It would be interesting to give the platform the possibility of using different meta-learning algorithms.

- **Dataset formats** Datasets can be stored in multiple formats. It would be interesting if the platform became able to recognize a considerable amount of formats.

- **Plug-ins** As seen before, a data miner has to spend a considerable amount of time in learning how to use a data mining tool. So it is understandable that for the most experience *miners*, a direct integration of the platform with their preferred tool might may seam more attractive. The development of plug-ins for the most common tools like Weka, Rapidminer, Knime, etc., may be a facilitator in making DM4D a more appealing object of interest.

- **Data preprocessing** Preprocessing is an important task in a data mining process because it can possibly reduce misleading results. However, this step can take a considerable amount of time and its the combination of tasks like cleaning, normalization, transformation, feature extraction, selection, etc. The integration of a preprocessing feature in DM4D could be an interesting addiction.

References

[AC98] Robert St Amant and Paul R Cohen. Intelligent Support for Exploratory Data Analysis. *Journal Of Computational And Graphical Statistics*, 7(4):545–558, 1998.

[Alt92] N S Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, 1992.

[Bas87] P.G. Bassett. Frame-Based Software Engineering. *IEEE Software*, 4(4), 1987.

[BDS08] D. Benslimane, S. Dustdar, and A. Sheth. Services Mashups: The New Generation of Web Applications. *IEEE Internet Computing*, 12(5), 2008.

[BEK+00] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. SOAP : Simple Object Access Protocol, 2000. Accessed on 30th September 2013.

[BGCSV08] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. 2008.

[BK07] M. Bachle and P. Kirchberg. Ruby on Rails. *IEEE Software*, 24(6), 2007.

[BPH05] Abraham Bernstein, Foster Provost, and Shawndra Hill. Towards Intelligent Assistance for a Data Mining Process: An Ontology-based Approach for Cost-sensitive Classification. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):133–518, 2005.

[BZ12] Paul C. Bryan and Kris Zyp. JSON Reference, 2012. Accessed on 30th September 2013.

[CDCS08] Michel Charest, Sylvain Delisle, Ofelia Cervantes, and Yanfen Shen. Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *Intelligent Data Analysis*, 12(2):211–236, 2008.

[CEF+06] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro, and W. Wang. Data mining curriculum: A proposal (Version 0.91). *Intensive Working Group of ACM SIGKDD Curriculum Committee. Retrieved March*, 25:2008, 2006.

[CG05] Grace Chung and Sara M. Grimes. Data mining the kids: Surveillance and market research strategies in children's online games. *Canadian Journal of Communication*, 30(4):527–548, 2005.

[CI84] Robert D. Cameron and M. Robert Ito. Grammar-based definition of metaprogramming systems. *ACM Transactions on Programming Languages and Systems*, 6(1):20–54, January 1984.

[Cle88] J.C. Cleaveland. Building application generators. *IEEE Software*, 5(4), 1988.

[Cro06] Douglas Crockford. The application/json Media Type for JavaScript Object Notation, 2006. Accessed on 30th September 2013.

[CW02] Krzysztof J. Cios and G. William Moore. Uniqueness of medical data mining, 2002.

[CZ10] Gary Court and Kris Zyp. A JSON Media Type for Describing the Structure and Meaning of JSON Documents. *IEFT*, 2010.

[DS10] L. Dusseault and J. Snell. RFC5789: PATCH Method for HTTP, 2010. Accessed on 30th September 2013.

[Eng96] R Engels. Planning tasks for knowledge discovery in databases; performing task-oriented user-guidance. *. . . th 2nd Int. Conf. on Knowledge Discovery in Databases*, 1996.

[FA05] John Fox and R Andersen. Using the R statistical computing environment to teach social statistics courses. *Department of Sociology, McMaster . . .* , (January), 2005.

[FGM+99] Roy Thomas Fielding, Jim Gettys, Jeff Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, 1999.

[Fie00] Roy Fielding. Representational state transfer. *Architectural Styles and the Design of Netowork-based Software Architecture*, pages 76–85, 2000.

[FMST94] C Feng, D Michie, D J Spiegelhalter, and C C Taylor. Machine Learning of Rules and Trees. In *Machine Learning Neural and Statistical Classificiation*, pages 50–83. Ellis Horwood, 1994.

[FN95] R T Fielding and H Frystyk Nielsen. Hypertext Transfer Protocol — HTTP/1.0. *Security*, (November 1994):1–57, 1995.

[Fox05] John Fox. The R Commander: A Basic-Statistics Graphical User Interface to R. *Journal of Statistical Software*, 14(9):1–42, 2005.

[FPSS96a] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases, 1996.

[FPSS96b] Usama M Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: an overview. In U M Fayyad, G Piatetsky-Shapiro, P Smyth, and R Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. American Association for Artificial Intelligence, 1996.

[GCFP+] Christophe Giraud-Carrier, Peter Flach, YongHong Peng, Jim Farrand, and Hilan Bensusan. METAL: A meta-learning assistant for providing user support in machine learning and data mining. Accessed on 20th July 2013.

[GJSB05] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification*. 2005.

[GP06] Brian Göetz and Addison Wesley Professional. Java Concurrency In Practice. *Building*, 39(11):384, 2006.

[Gun02] Berton H Gunter. S Programming, 2002.

[Han97] DJ J J Hand. Intelligent data analysis: Issues and opportunities. *... in Intelligent Data Analysis Reasoning about Data*, pages 1–14, 1997.

[Han03] David Heinemeier Hansson. Ruby on Rails, 2003. Accessed on 16th September 2013.

[HF05] David Heinemeier Hansson and Thomas Fuchs. *Agile Web Development with Rails*. Pragmatic Bookshelf, 2005.

[HG08] Jiawei Han and Jing Gao. Research challenges for data mining in science and engineering. In *Next Generation of Data Mining. Chapman & Hall/CRC*, pages 1–18. 2008.

[IG93] Ross Ihaka and Robert Gentleman. R programming language, 1993. Accessed on 16th September 2013.

[Jam01] David A James. An R / S Interface to the MySQL Database. Technical report, 2001.

[KBF+00] Ron Kohavi, Carla Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. KDD-Cup 2000 Organizers' Report: Peeling the Onion. *SIGKDD Explorations*, 2(2):86–98, 2000.

[KH01] A Kalousis and M Hilario. Feature selection for meta-learning. *Advances in Knowledge Discovery and Data ...*, 2001.

[KR88] Brian W Kernighan and Dennis M Ritchie. *The C programming language*, volume 78. Prentice Hall, 1988.

[LL10] Michael Lawrence and Duncan Temple Lang. RGtk2: A graphical user interface toolkit for R. *Journal of Statistical Software*, 37(8):1–52, 2010.

[Man10] Mysql Reference Manual. MySQL 5.0 Reference Manual. *Syntax*, page 3079, 2010.

[Mat95] Yukihiro Matsumoto. Ruby programming language, 1995. Accessed on 16th September 2013.

[Mil09] Alex Miller. Core Java Concurrency. *RefCardz*, pages 1–6, 2009.

[MO07] C Mesnage and Eyal Oren. Extending ruby on rails for semantic web applications. *Web Engineering*, pages 506–510, 2007.

[MR10] OZ Maimon and L Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer US, 2010.

[MR11] Ralf Mikut and Markus Reischl. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):431–443, 2011.

[MS04] K Morik and M Scholz. The MiningMart Approach to Knowledge Discovery in Databases. *Intelligent Technologies for Information Analysis*, pages 47–65, 2004.

[MST94] Editors D Michie, D J Spiegelhalter, and C C Taylor. Machine Learning , Neural and Statistical Classification. *Technometrics*, 37(4):459, 1994.

REFERENCES

[Mue] Robert Muenchen. The Popularity of Data Analysis Software. Accessed on 16th September 2013.

[NT95] Ikujiro Nonaka and Hirotaka Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, volume 29. Oxford University Press, 1995.

[OVK+03] OBJECT MANAGEMENT GROUP, M D A Guide Version, Alan Kennedy, Kennedy Carter, and William Frank X-change Technologies. MDA Guide Version 1.0.1. *Object Management Group*, 234(June):51, 2003.

[RC86] J Rees and W Clinger. Revised report on the algorithmic language scheme, 1986.

[Ric75] JR R R Rice. The algorithm selection problem. pages 75–152, 1975.

[Sch94] C Schaffer. A conservation law for generalization performance. *Proceedings of the Eleventh International Conference . . .* , 1994.

[Sch06] D C Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *Computer*, 39(2):25–31, 2006.

[Spe04] Charles Spearman. Spearman ' s rank correlation coefficient. *Amer. J. Psychol.*, 15:72–101, 1904.

[SRC+95] D. Sleeman, M. Rissakis, S. Craw, N. Graner, and S. Sharma. Consultant-2: pre- and post-processing of Machine Learning applications. *International Journal of Human-Computer Studies*, 43(1):43–63, July 1995.

[SVB12] Floarea Serban, Joaquin Vanschoren, and Abraham Bernstein. A survey of intelligent assistants for data analysis. *ACM Computing . . .* , V(212):1–35, 2012.

[Tiz08] Tizag. MySQL Tutorial, 2008.

[TSD09] Qiming Teng Qiming Teng, P.F. Sweeney, and E. Duesterwald. Understanding the cost of thread migration for multi-threaded Java applications running on a multicore platform. *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 2009.

[Ver09] Carlo Vercellis. Business Intelligence : Data Mining and Optimization for Decision Making. In Carlo Vercellis, editor, *Business Inteligence: Data mining and optimization for decision making*, chapter Book front, page 16. Wiley, 2009.

[WBK06] Siri Krishan Wasan, Vasudha Bhatnagar, and Harleen Kaur. The impact of data mining techniques on medical diagnostics, 2006.

[WM97] D H Wolpert and W G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[WM09] B.G. Weber and M. Mateas. A data mining approach to strategy prediction. *2009 IEEE Symposium on Computational Intelligence and Games*, 2009.

[WN13] Michael Whittaker and Jürgen Neumann. Develop in the fast lane with Ruby/Rails. *Software Developer's Journal*, 2(3):10–14, 2013.

[Wol95] DH Wolpert. No free lunch theorems for search. *Most*, pages 1–38, 1995.

[Wol01] David H Wolpert. The supervised learning no-free-lunch theorems. *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 6(1):1–20, 2001.

[WPM+09] Tobias Wrigstad, Filip Pizlo, Fadi Meawad, Lei Zhao, and Jan Vitek. Loci : Simple Thread-Locality for Java. *ECOOP*, pages 445–469, 2009.

[XTLL09] Tao Xie, Suresh Thummalapenta, David Lo, and Chao Liu. Data Mining for Software Engineering, 2009.