

A Rule-Based Data Mining Assistant

Roubini Xenou

Thesis submitted in partial fulfillment of the requirements for the

Master of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department

Thesis Advisors: Prof. *Ioannis Tsamardinos*

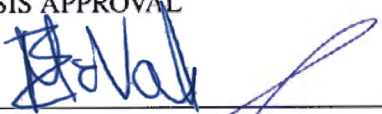
UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

A Rule-Based Data Mining Assistant

Thesis submitted by
Roubini Xenou
in partial fulfillment of the requirements for the
Master of Science degree in Computer Science

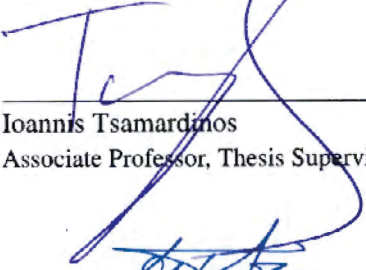
THESIS APPROVAL

Author:

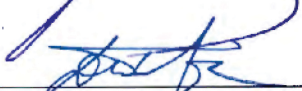


Roubini Xenou

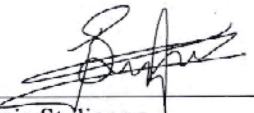
Committee approvals:



Ioannis Tsamardinos
Associate Professor, Thesis Supervisor




Dimitris Plexousakis
Professor, Committee Member



Yannis Stylianou
Professor, Committee Member

Departmental approval:



Antonis Argyros
Professor, Director of Graduate Studies

Heraklion, January 2017

Abstract

Data mining, as well as Data Analysis and Machine Learning are utilized during the last few years, by a variety of people of different experience level and from different fields, to provide solutions in a variety of different domains and applications. Machine Learning algorithms are applied to any dataset, regardless of the type or content, and regardless of source e.g. real world or simulated, observations, and can generate a model describing them.

The available data mining algorithm and methodologies space, is increased day by day, making it difficult even for experts to follow this changes. As different dataset types may demand, a different approach in terms of methodology or algorithmic analysis, the Data Analysis process is becoming even more complex. A lot of effort has been given towards the development of Data Mining Assistants (usually referred as IDAs - Intelligent Data Assistants), in order to overcome the above obstacles.

In this Thesis we designed and developed an automated intelligent system, the RB-DMA (Rule Based Data Mining Assistant), which, based on an extension of the OntoDM data mining ontology [1] and combined with a set of rules written in Drools [2], proposes the most appropriate data mining workflows, ranked based on their efficiency for a given analysis.

Our approach provides, all the decisions the end-user will need, regardless of their experience or knowledge, in order to conduct an analysis with trustful results.

Data Analysis depending on the amount and complexity of data, usually require a considerable amount of time in order to produce results. Our system, addresses this issue, by proposing to the user the n best workflows, where n is considerably small and optimized to produce close to best results.

Last, but not least, the system covers up to 200 data analysis scenarios (binary classification and regression, on a variety of data types and sizes)..

Περίληψη

Η ανάλυση των δεδομένων είναι μια αναδυόμενη και σε κάθε περίπτωση χρήσιμη επιστήμη. Οι διάφοροι αλγόριθμοι μηχανικής μάθησης παρέχουν τη δυνατότητα να εκπαιδεύονται σε ένα σύνολο δεδομένων, αποτελούμενο από κάθε είδους, πραγματικές ή προσομοιωμένες, παρατηρήσεις, και να δημιουργούν ένα μοντέλο που τα περιγράφει. Με το πέρασμα του χρόνου, ο αριθμός των περιπτώσεων, σε επαγγελματικό ή καθημερινό επίπεδο, που διευκολύνεται από την ανάπτυξη μιας μεθόδου εξόρυξης δεδομένων γίνεται όλο και μεγαλύτερος. Προκειμένου να καλύφθούν οι πολυποίκιλες ανάγκες που προκύπτουν, ο χώρος των διαθέσιμων αλγορίθμων και μεθοδολογιών εξόρυξης δεδομένων ολοένα αυξάνεται, καθιστώντας την εξερεύνηση τους ως μια επίπονη και χρονοβόρο διαδικασία, ακόμη και για τους πιο πεπειραμένους αναλυτές δεδομένων. Μία ακόμη δυσκολία, είναι η απαίτηση ξεχωριστής κατάλληλης μαθοδολογίας και ερμηνείας για κάθε διαφορετικό τύπο δεδομένων. Μια μεγάλη προσπάθεια έχει δοθεί στην ανάπτυξη καθοδηγητών εξόρυξης δεδομένων, με σκοπό να βοηθήσουν το χρήστη να ξεπεράσει τα παραπάνω εμπόδια. Μέχρι στιγμής, οι καθοδηγητές ταξινομούνται ως αυτοματοποιημένοι και συνεργατικοί. Στην εργασία αυτή, σχεδιάστηκε και αναπτύχθηκε ένα αυτοματοποιημένο έξυπνο σύστημα, το **RB-DMA**, το οποίο, βασισμένο στην **OntoDM** οντολογία, σε συνδυασμό με ένα σύνολο κανόνων εκφρασμένων με την βοήθεια του συστήματος *drools*, προτείνει τις πιο κατάλληλες ροές εργασιών εξόρυξης δεδομένων, διατεταγμένες με βάση την αποτελεσματικότητά τους για μια δεδομένη ανάλυση. Η προσέγγισή μας παρέχει, σε χρήστες οποιουδήποτε επιπέδου γνώση, όλες τις αποφάσεις που χρειάζονται προκειμένου να προβούν σε μία ανάλυση με αξιόπιστα αποτελέσματα. Για έναν χρήστη με πλήρη άγνοια, η ανάγκη να γίνει έμπειρος' εξαλείφεται. Από την άλλη πλευρά, το σύστημα θα λειτουργεί περισσότερο ως ένας μηχανισμός υπενθύμισης των διαθέσιμων βέλτιστων πρακτικών για τον εμπειρο χρήστη. Ακόμη, το σύστημά μας βοηθάει στην μείωση του απαιτούμενου χρόνου για να πραγματοποιηθεί μία ανάλυση, εγγυόμενο, στις περισσότερες περιπτώσεις, σχεδόν βέλτιστα αποτελέσματα εκτελώντας μόνο τις πρώτες *K* καλύτερες ροές. Τελευταίο, αλλά όχι λιγότερο σημαντικό, το σύστημα καλύπτει έως 200 σενάρια ανάλυσης δεδομένων (ταξινόμηση δύο κλάσεων, και παλινδρόμηση με διαφορετικούς τύπους και μεγέθη δεδομένων), βοηθώντας των αναλυτή να ξεπεράσει το προαναφερθέν πρόβλημα της ξεχωριστής διαχείρισης διαφορετικών δεδομένων.

Acknowledgements

First of all, I would like to thank my supervisor, Professor Yannis Tsamardinos, for giving me the opportunity to become a member of his team, for his guidance and trust that he showed during the time we worked together.

I would also like to thank all my colleagues at the Laboratory. Thank you Giorgos A. , Lisa, Michalis, Maria, Giorgos Mp, Paulos, and all the others!

Of a great help were the long discussions with Dr. Theodore Patkos, member of the Information Systems Laboratory - FORTH. I owe him, at least, a great Thank you!!

I would also like to thank my closest friends Kaiti, Dora, and Elena, for their love, trust and support all these years.

Last but not least, the greatest “thank you” to my family: my parents, Panagiotis and Eutuxia, for their love, trust, encourage and patient during the years of my studies. My sister Marilena for showing so much patient and love. Finally, but most important, I would like to thank my grandmother Maria for taking good care of me during all these years, living in the same house!

Thank you all!

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Motivating Example	3
1.3	Organization of this Thesis	5
2	Related Work	9
2.1	CITRUS	9
2.2	Similarity Matching Approach	9
2.3	Generating and Scoring workflows using meta-mining	10
2.4	IDEA	10
2.5	PDM	11
3	Implementation	13
3.1	The employed Ontologies	13
3.1.1	OntoDM Ontology	13
3.1.2	Incorporating and Extending OntoDM	15
3.2	The systems architecture	17
3.3	Scoring workflows using heuristics	21
3.3.1	The scoring function	21
3.3.2	The scoring heuristics	22
4	Evaluation	33
4.1	The datasets analyzed	33
4.2	Evaluating the system	33
4.3	Evaluating the proposed workflow instances	34
4.3.1	Data analysis results	34
5	Conclusions and Future Work	47

List of Figures

1.1	The phases of a data mining workflow, considering the standard framework developed in CRISP-DM [3]	2
1.2	The five first steps executed by the system in order to identify the required data mining task (step 1, green arrows), and obtain the compatible components for the different workflow phases (step 2 - 5, brown, purple, pink, and black arrows). The thin dashed arrows denote the repeated execution of the get-Hyperparameter sub-steps. The knowledge base consists of a data mining ontology, and the rule mechanism is a set of heuristics related to the data mining entities. Following the different color paths one can see the process executed during each of the five steps.	6
1.3	The sixth and last step executed by the system, in order to compose the proposed workflow instances for a dataset and rank them, considering also user preferences	7
3.1	The main structure of the BFO and IAO ontologies, which OntoDM is extending	14
3.2	The structure of the OntoDM entities mainly used in RB-DMA. OntoDM:output data specification represents the target type of a dataset, OntoDM: Algorithm implementation represents an algorithm implementing a data mining algorithm, OntoDM: parameter represents the hyperparameters of a data mining algorithm implementation. The bold lines represent sub-ClassOf relations, showing which are the OBI, BFO, and IAO extended classes.	15
3.3	OntoDM extension, the red font rectangles represent the newly created entities. Data preprocessing algorithm, Dimensionality reduction algorithm, and Imputation algorithm are extending the OntoDM: Data processing algorithm, in order to represent the algorithms of the respective workflow phases. Imputation is part of the data preprocessing phase, but is represented as a separate entity. The entities represented as subclasses of the OntoDM: Algorithm Implementation entity are the algorithms implementing the respective workflow phases.	16

3.4	rbf kernel support vector machine (r_svm) use case example: Assuming a dataset with binary target (represented by OntoDM: boolean output), the system following the Has part relation, identifies a binary classification task (represented by OntoDM: binary classification task). Starting from the binary classification task entity, and following the Has part relation, it concludes that binary classification algorithms are the appropriate modeling components. Through the isConcretizationOf relation, the rbf kernel svm is returned as an implementation of a binary classification algorithm. Finally, the Has quality relation is being used, in order to identify the methods' appropriate tuning hyperparameters, gamma and cost, which are represented as binary r_svm parameter instances.	18
3.5	The system contains two main mechanisms, the knowledge base, which consists of data mining entities represented in the OntoDM ontology, and the rule mechanism, a set of rules expressed using the Drools system. The system executes 6 steps in order to compose and rank the proposed workflow instances. The process of the first step is represented through the arrow denoting the dataset given as input to the knowledge base, and the DM_task arrow, which represents the identification of the appropriate data mining task, after reading the dataset. The arrow connecting the knowledge base to the rule mechanism, represents the possible components of the workflow instances for each of the four workflow phases (i). The arrows labeled as ranked_Si_components, represent the ranked and filtered workflow components, according to the rules of the rule mechanism. In the last step the workflow composer and ranker takes as input all the possible methods for each workflow phase, retrieved during the previous 4 steps, composes the appropriate workflow instances, and finally ranks them.	19
3.6	Best performing cost and gamma configurations, according to scikit-library documentation's experiments.	24
4.1	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Bernstein datasets. . . .	36
4.2	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Vsmall,small sample sizes Classification, (* = <i>dataset with missing values</i>)	37
4.3	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Medium sample size Classification, (* = <i>dataset with missing values (NaN)</i>)	42

4.4	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Large sample size Classification	43
4.5	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Vsmall,small sample sizes Regression	44
4.6	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Medium sample size Regression	45
4.7	plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Large sample size Regression	46

List of Tables

3.1	The relations used to represent the data mining entity qualities in the explanation column.	17
3.2	Table of general support vector machine heuristics. Cost is an svm tuning hyperparameter, representing the svm penalty factor. Epsilon is another svm hyperparameter, used mainly when training regression models, affecting the support vectors quantity. AvgAccuracy is a metric, defined in our system, which represents the average performance capability of an algorithm, taking values in the range of 1 - 5, where 1 is the lowest performance capability, while 5 is the highest one	22
3.3	Table of linear kernel svm heuristics. AvgAccuracy \in [1 - 5].	22
3.4	Table of rbf kernel svm heuristics. The condition $c == \text{median}(\text{range})$ denotes, cost equal to the median of cost range. Gamma is a tuning hyperparameter for rbf and polynomial kernels, controlling the area of the support vectors influence. AvgAccuracy \in [1 - 5].	23
3.5	Gamma is a tuning hyperparameter for rbf and polynomial kernels, controlling the area of the support vectors influence. AvgAccuracy \in [1 - 5].	23
3.6	Table of elastic-net heuristics. Alpha is the elastic-net mixing hyperparameter, controlling the weighted combination of L1 and L2 penalties of the model. Lambda is the hyperparameter controlling the amount of feature coefficients shrinkage. AvgAccuracy \in [1 - 5].	25
3.7	Table of Random Forests heuristics. NumPredictorsToSample is the number of features to select at random for each decision split, for a decision tree of a random forest. Minleaf, controls the minimum accepted size for the leafs of a decision tree. FeatNum denotes the dataset dimensionality. AvgAccuracy \in [1 - 5].	26
3.8	Table of knn heuristics. K is the hyperparameter controlling the amount of neighbors considered in the knn model. AvgAccuracy \in [1 - 5].	27
3.9	Table of Decision Trees heuristics. AvgAccuracy \in [1 - 5].	27
3.10	Table of ses heuristics. Maxk sets the maximum conditioning set to use in the multiple conditional independence tests performed. Threshold is the parameter used for assessing the p-value significance in the independence tests. AvgAccuracy \in [1 - 5].	27
3.11	Rank of interpretability for the 5 modeling methods	28

3.12	Train and Predict Complexity of the methods (used as ranking heuristic). S represents the sample size, F represents the feature size, k , T , and $maxk$ are the selected hyperparameter values for the respective algorithms.	28
3.13	Preprocessing Stage heuristics. NumFeat represents the dataset dimen- sionality.	29
3.14	The Model Selector and Performance Estimator stage heuristics. The evaluation of the heuristics conditions is executed sequentially. The pro- cess stops when a condition evaluates to true. Diff equals $ SampleRank -$ $FeatureRank $, for example small - vsmall = 1. $\min(ClassSize)$ repre- sents the size of the dataset's smallest class. TS stands for Train Set, and VS stands for Validation Set.	30
3.15	Sample size ranks	30
3.16	Feature size ranks	30
4.1	Table presenting the datasets analyzed to evaluate the system. The datasets having underscore and a number, concatenated to their name are subsam- ples of the original ones. The last column indicates those that were also analyzed by <i>Bernstein et.al</i> [4]. (vsmall and vlarge refers to very small and very large sizes, respectively).	39
4.2	Time needed to generate the workflow instances for 11 different datasets, evaluated in seconds	40
4.3	Pearson's r_s between the system's workflow instance ranking, and the actual one, for the classification task datasets. Br_s column depicts the correlations resulted in <i>Bernstein et.al</i> [4] for the same datasets.	40
4.4	Comparing IDEA and RB-DMA mean - median Pearson's r_s	41
4.5	Pearson's r_s between the system's resulted workflow instance ranking, and the actual one, for the regression task datasets.	41

Chapter 1

Introduction

1.1 Motivation

In this era data analysis is considered to be applicable and provide solutions in a variety of different domains. Machine Learning algorithms can be trained on any kind of dataset, real world or simulated, and generate a model describing their structure. These models can be used for the classification of future data instances or/and the extraction of conclusions regarding the investigated subject.

Nowadays machine learning is being used in everyday situations, to make future decisions, to predict, to prevent malicious events and also in automation. In order for data analysis tools to produce good quality results, those who will use it need to have, at least, some basic knowledge and understanding of the algorithms, pipelines and methodologies that are used in machine learning.

As the algorithms and the methodologies are enriched more and more, even senior data analysts find it difficult to keep up with the pace, and select the most appropriate and effective methods in each case.

There are also cases, that due to time or resource limitations, the application and comparison of different methodologies upon the same dataset, in order to find the best performing one, is prohibited.

Another reason why Data Analysis can be a challenging and difficult process, is the quality of the data itself. Datasets can vary in the form and the content. A dataset may consist of categorical, numerical, binary, or mixed data instances. Also, each data instance may be annotated or not annotated. There can be cases that the data are incomplete, in the sense that feature values for data instances are missing. All of the above, lead to the conclusion that separate interpretation may be needed, in each case.

In order to resolve these issues, a lot of effort has been given to create intelligent automated systems known as IDA (Intelligent Data Mining Assistants) that can help data analysts to choose and apply the appropriate data mining workflow instance. The IDAs can either be “automated” or “cooperative”. Automated IDAs are the systems that will compose efficient workflow instances without the need of any interaction by the user. [5], [4], [6]. Cooperative IDAs are the systems that guide the user, step by step, by asking for

her feedback throughout the whole procedure. [7], [8].

In this Thesis, we designed and developed an IDA system which uses an extended version of the OntoDM ontology [1] and a set of rules (heuristics), written in Drools [2]. The use of the above technologies, make it feasible to express any kind or amount of heuristics (rules), annotate semantically the provided dataset and propose a series of appropriate workflow instances. These workflow instances are ordered based on their estimated efficiency in each case.

A workflow, according to CRISP-DM, 2000 [3], is organized into a small number of top level phases. Figure 1.1 depicts an idealized sequence of discrete steps (phases / tasks) (preprocessing, feature selection, model, model selector & performance estimator, more detailed explained in chapter 3, section 2), performed according to the indicated order. In practice, many of the tasks may be omitted, or executed in a different order. Sometimes it may be necessary to backtrack to previous tasks and repeat certain actions.

Our approach provides the user with all the decisions he needs in order to conduct an analysis, and extract reliable results without the need of much effort or experience.

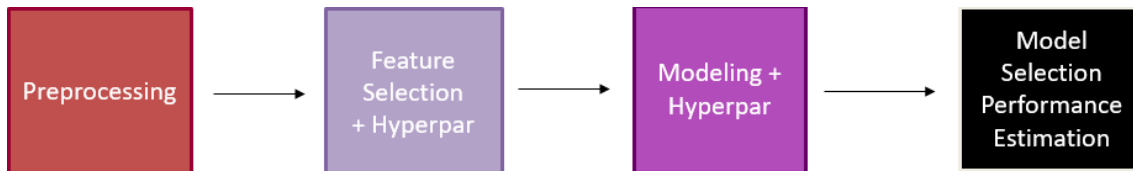


Figure 1.1: The phases of a data mining workflow, considering the standard framework developed in CRISP-DM [3]

Our implementation provide, if needed, the ability to the user to freely intervene in the proposed pipelines before using them in his preferred platform and implementation. It acts as a knowledge base repository that the user can ask for advices in order to achieve the most efficient solution.

The system can easily be used by both experienced and inexperienced users.

Our approach offers the ability to safely execute only the first K ($K \ll 0.5 * \text{NumWorkflows}$) from the proposed series of workflow instances, assuring that results closest to best will be obtained. This allows the user to override time limitations, as he can achieve the same or better result in less time. For the rest of the Thesis we will refer to this claim as "the first K workflows" claim.

Furthermore, the user can select which weights he wants the system to use, choosing between accuracy, time and comprehensiveness. This way he can customize, based on his own needs and limitations, the obtained series of workflow instances.

Our implementation covers a wide range of 200 data analysis scenarios. More detailed, our system is capable of handling 25 different sample size, and feature size dataset categories, classification and regression datasets (2), datasets with and without missing values (2), and also datasets with continuous and categorical feature type (2). Summing up, the total number of different dataset categories, which our system is capable of handling equals to $Total = 25 * 2 * 2 * 2 = 200$. The large number of cases covered by our system,

is a fact that enables the data analyst to deal with a diversity of datasets, without the need of separate prior understanding and interpretation.

To summarize, our system intends to act as an analyst itself. It collects all the knowledge related to the data analysis tasks in one place, and continuously leverages it in order to suggest decisions. Thus, the system reduces the effort and time the user would need to spend, while providing more reliable and comprehensive results.

In order to test the confidence and performance of our system, we proceeded with the analysis of 31 datasets of binary classification and regression tasks from multiple sources and fields, with different data sizes, using the proposed by our system pipelines. The results are presented in this document.

1.2 Motivating Example

RB-DMA (Rule-Based Data Mining Assistant) produces a series of data mining workflow instances for a given dataset, which implies a data mining task, taking also into consideration the additional user preferences (i.e: accuracy vs time, comprehensibility vs accuracy or time vs comprehensibility). We are briefly presenting the generated workflow instances for a real world dataset.

Consider the case where one wants to analyze these data [9]. The dataset is related with direct marketing campaigns of a Portuguese banking institution, based on phone calls. The outcome is a binary, "yes" or "no", trying to predict if a client will subscribe a term deposit. Each data instance refers to one client and is composed by 17 features describing some of his personal information.

The figures 1.2, and 1.3 are depicting the series of steps executed by the system in order to construct the appropriate data analysis workflow instances. The light blue rectangle in fig 1.2 represents the main system, which is responsible for the coordination of the whole process, and also for the communication between the knowledge base and the rule mechanism, represented by the beige and dark red rectangles, respectively. The dashed blue vertex between them, implies an indirect connection through the main system. All the important and useful information and components for conducting the bank dataset analysis (i.e DM techniques, algorithms, hyperparameters e.t.c.) are being stored into the knowledge base. The rule mechanism is responsible for filtering and/or ranking the obtained information (i.e the data mining workflow components), and returning the final components for the composition of the appropriate workflow instances. More precisely, the execution steps are six (thoroughly described in section 3.2)

1. "Requested task identification"
2. "Preprocessing"
3. "Feature Selection"
4. "Modeling algorithm"

5. "Model Selection and Performance Estimation"
6. "Final Workflow Composition and Ranking"

There is also one more sub-step executed as part of the Preprocessing and Feature Selection main steps, the "hyperparameter tuning". The colored vertices in the figure are representing the different execution steps (for the steps 2 - 5 the colors used are the same as in Figure 1.1)

The green path represents the "Requested task identification" step during which,

1. The task is retrieved from the knowledge base based on the dataset
2. The returned task is passed as input to the knowledge base for the next steps (represented using the double arrow)

In order to identify the preprocessing needs of the dataset the brown colored path is executed, during which

1. The Preprocessing methods are retrieved from the knowledge base
2. Both data size and feature type are passed as input to the rule mechanism
3. The answer is filtered through the rule mechanism (that keeps only those necessary for the dataset)
4. The variable `fs:true` is passed to the main system in order to enable the execution of Feature Selection step

Following the purple vertices path we can see that, during the Feature Selection step, the system:

1. The Feature Selection methods are retrieved from the knowledge base
2. The results are ranked through the rule mechanism
3. For each FS method returned by (1) && (2):
 - (a) The Hyperparameters are retrieved from the knowledge base
 - (b) The results are ranked through the rule mechanism
 - (c) The respective Preprocessing configurations are returned from the composer i.e Elastic-net conf 1 ...

The dashed arrows represent the recursiveness of steps 3a, 3b, and 3c.

The same exact process is also executed for the Modeling step, represented by the pink vertices path.

In order to obtain the Model Selection and the Performance Estimation components, which are the last of the workflow, RB-DMA executes the black arrow path which,

1. Takes into consideration the bank data size which is already passed to the rule mechanism
2. The rule mechanism outputs, “three-fold cross validation” and “hold out” as model selection and performance estimation methods, respectively

The last step, which is depicted in figure 1.3 is represented as the purple rectangle "workflow composer and ranker" and it wraps three different system components. The composer and rule mechanism, and a ranking function giving the final rank of the composed workflows, taking into consideration the user preferences.

Our system autonomously leverages all the available information (knowledge base) and rules (rule mechanism) in order to take complex decisions, similar to those listed in the example above, and automatically provide to the user the final constructed workflow instances.

1.3 Organization of this Thesis

The rest of the Thesis is structured as follows. Chapter 2 briefly describes work that has been done related to Intelligent data mining systems. Following, section 3.1 describes the OntoDM ontology and our extension. In 3.2 and 3.3 the architecture of the system and the workflow ranking heuristics and methodology are presented, respectively. Finally, in the last two sections we describe the evaluating methodology that we followed, present the results of the evaluation, and conclude.

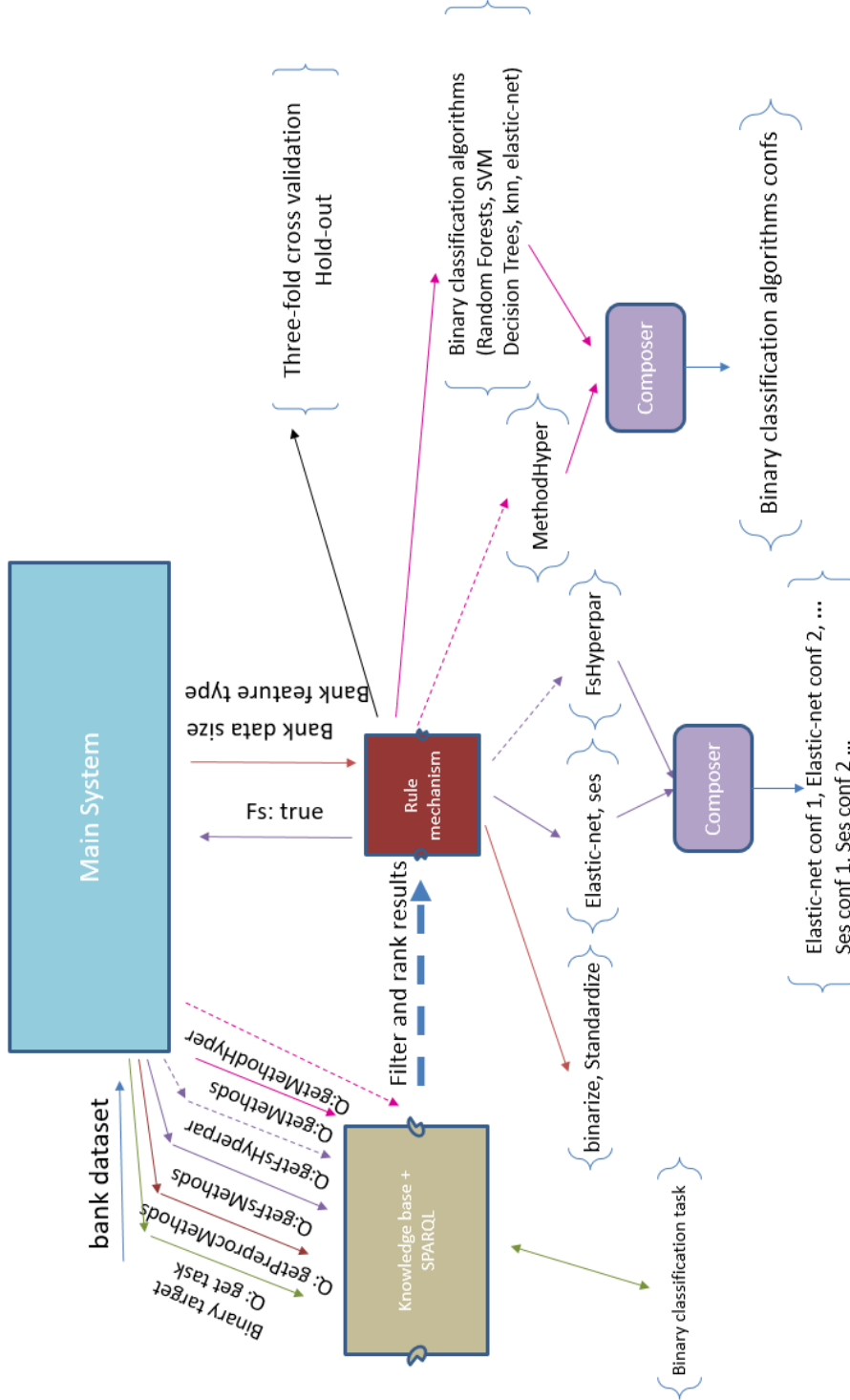


Figure 1.2: The five first steps executed by the system in order to identify the required data mining task (step 1, green arrows), and obtain the compatible components for the different workflow phases (step 2 - 5, brown, purple, pink, and black arrows). The thin dashed arrows denote the repeated execution of the get-Hyperparameter sub-steps. The knowledge base consists of a data mining ontology, and the rule mechanism is a set of heuristics related to the data mining entities. Following the different color paths one can see the process executed during each of the five steps.

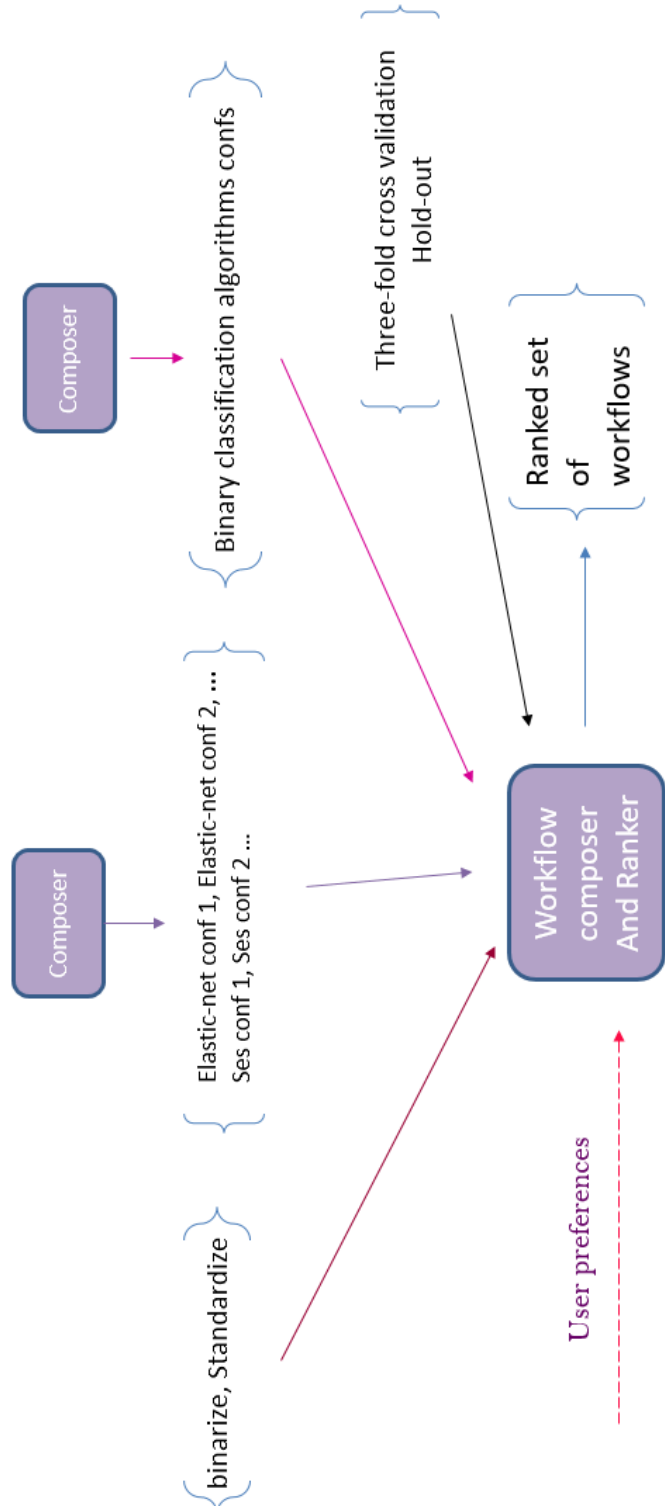


Figure 1.3: The sixth and last step executed by the system, in order to compose the proposed workflow instances for a dataset and rank them, considering also user preferences

Chapter 2

Related Work

During the recent years, a lot of effort has been spent towards the development of efficient Data Mining Assistants. The currently existed ones are separated in two categories, the automated, and the cooperative assistants (as reviewed in [10]). The automated, automatically generates and provides a list of valid workflows instances. The cooperative, guides and cooperates with the user through the whole process of the workflow instance composition, taking into account his preferences. Both of them, employ a planning algorithm which, relying on meta-learning and/or a data-mining Ontology, composes the appropriate workflow instances, based on available DM Method's preconditions and postconditions.

2.1 CITRUS

CITRUS [7] was the first full step IDA, (guiding the user through the whole procedure of the workflow instance composition) and was designed to perform as a cooperative assistant. The system initially performs a top level decomposition of the KDD-task into subtasks. A recursive refinement of the subtasks is accomplished from a PSM library (Problem Solving Methods) which finally, by exploiting data characteristics (i.e standard deviations, attribute types, missing values), it ends up with the proposed algorithms. Tasks are mapped to the proper PSM using same concept preconditions and postconditions. This approach forms a kind of hierarchical task planning system.

2.2 Similarity Matching Approach

Diamantini *et al.* in [5] introduced a Similarity matching approach of a cooperative assistant which is relying on KDDONTO Ontology [11]. Their system goes backwards iteratively, starting from the given task, adding the appropriate algorithms for each workflow stage. The quality of each proposed algorithm depends on a similarity matching function ensemble with the algorithm's performance. Processes are ranked using this quality assessment, so that users are provided with a criterion to choose the most suitable with respect to their requests. This system, in a similar way that our system does, is less strict on the process composition (unlike [8], [7] in Hierarchical Task Network (HTN)

decomposition), considering that any algorithm could possibly match the previous one by adding the appropriate preprocessing steps. Moreover, their approach, just like our own RB-DMA, is aimed to achieve abstract and reusable KDD prototype processes, that can be seen as suggested sequences of the conceptual steps.

2.3 Generating and Scoring workflows using meta-mining

eProPlan [8] is designed to perform mainly as a cooperative assistant but also can be used as an automated workflow instance generator. The system, similarly to CITRUS, carry out an initial decomposition of the user-specified task into sub-steps, which can be further decomposed afterwards, through a GUI. The implementation depends on a HTN planner combined with the DMO ontology which consists of the DMOP (Data Mining Optimization Ontology)[12] and the DMWF (Data Mining Workflow Ontology). Alternative execution plans are generated, composed by the several methods available for a given (sub)task. The user is provided with the set of the workflow instances and makes the final decision. What is more, compared to CITRUS, is that they are making the hierarchical planning a part of their ontology. They also take into account each data attribute separately into their final proposed algorithms. The system lacks of a method for ranking the proposed workflow instances in order to guide the user better. Automated experimentation is a viable but yet computationally expensive solution.

For this purpose, later on, [13] Nguyen *et al.* implemented a meta-mining approach which, for the first time, move meta-learning one step forward. In contrast to other approaches they stop considering learning algorithms a “black box”. More precisely, they employ as predictors, not only the features of the previously analyzed datasets (meta-learning), but also characteristics of the algorithm, such as the model structure, the objective functions and search strategies used, or the type of data partitions produced,(meta-mining). By doing so they manage to generalize over the modeling algorithms. This method is used in combination with eProPlan in order to label the proposed workflow instances as best or rest. One drawback of this system is that it has to analyze a respectful number of datasets with various different workflow instances and store the resulted datasets in order to become efficient. This is an expensive process in terms of time and computations. Also, the reliability of such a ranking system is tightly connected to the efficiency of the trained meta-models.

2.4 IDEA

IDEA(Intelligent Discovery Electronic Assistant) [4], [14] is an approach really close to ours. The System relies on an Ontology of Data Mining Operators which also expresses some preconditions and postconditions of them. The Knowledge Discovery process is being segmented into three consecutive (sub)processes, pre-processing, induction, and post-processing. In addition to them, our own RB-DMA (Rule Based Data Mining Assistant) moves one step forward and wraps the workflow segments into a Model Selector

combined with a Performance Estimator which are both appropriately selected for the current dataset and are treated as parts of the workflow. Moreover, we also incorporate the method's hyper-parameters tuning as part of the induction step.(the induction segment is further decomposed into two smaller (sub)processes, the model training and the method's hyperparameters tuning). The IDEA acts as an assistant to the user, as it provides a ranked list of all the potentially valid processes for his task. An AI-planner uses the restrictions and effects (preconditions and postconditions) of applying each operator, in order to compose appropriate workflow instances for a given task and dataset. Bernstein *et al.* [14] are using heuristics, in the same way we do, in order to offer the capability to rank the processes by many different aspects, such as speed, accuracy, cost, sensitivity, comprehensibility and weighted combinations of the above components. This approach has the restriction of defining all the heuristics inside the ontology and only uses the internal inference engine, a fact that limits the expressiveness. In our approach, except from the ontology we also use an external rule engine which enables us to express unobtrusively, to add and combine easily heuristics and intuitions related to each of the (sub)-processes.

2.5 PDM

PDM [6] describes KDD operators, their preconditions, and effects, by defining a planning domain in PDDL (Planning Domain Definition Language)[15]. The cost effects (i.e time complexity, accuracy) are computed based on a meta-learning technique. It uses the DM standard language PMML [16], to describe KDD tasks. The system receives as input a KDD task, specified in a PMML file, which is automatically translated into a planning problem described in PDDL. Any planner can be used to generate a plan (or plans) which are then translated into DM workflows .Thus, it is considered as an automated workflow generator. Similarly to our approach, they incorporate a Model Selector as a part of their workflows, but yet they are only using default values as hyperparameters.The system is relying on meta-learning rules which, as already mentioned, is not considered as an advantage. Finally, the system lacks the potential of ranking the workflow instances by different (weighted) factors.

Chapter 3

Implementation

3.1 The employed Ontologies

3.1.1 OntoDM Ontology

OntoDM [1] is a Heavy-Weight Ontology describing the outcomes of DM (Data Mining) research, and thus covering completely the domain, and supporting more intelligent data mining methods. On the contrary, most of the other efforts are mostly designed to deal with a specific task and thus are characterized as Light-Weight ontologies. The OntoDM Ontology is designed with a sound theoretical foundation and intends to share and reuse existing knowledge. For this purpose, the top-level classes of BFO ¹, (depicted in figure 3.1), IAO ², and OBI ³ ontologies, are being extended by data mining specific classes. It also uses the relations of the OBO ⁴ and RO ontologies in order to ensure completeness and single inheritance for all entities. All the above form good practices in ontology development field.

In figure 3.1 the main structure/entities of the IAO, BFO and OBI Ontologies are depicted, together with the relations that express their in-between relationships. The bold arrows denote `rdfs:subClassOf(+)` (is-a) relations. The entities **Specifically Dependent Continuant** (SDC) and **Generically Dependent Continuant** (GDC) are connected through **IAO:isConcretizationOf** relation. The descendants of SDC and GDC, OBI:plan and IAO:plan specification, are also connected through the same relation. An **IAO:plan specification has_part** an **IAO: objective specification**. Finally, BFO:Quality entity which represents the is-a SDC.

As a result, OntoDM is an advantageous ontology which exploits the above characteristics, in order to improve the KDD process. The main super-classes (entities) of OntoDM are:

- data specification

¹BFO: <http://www.ifomis.org/bfo>

²IAO: <http://code.google.com/p/information-artifact-ontology/>

³OBI: <http://purl.obolibrary.org/obo/obi>

⁴OBO: http://ontoworld.org/wiki/OBO_foundry

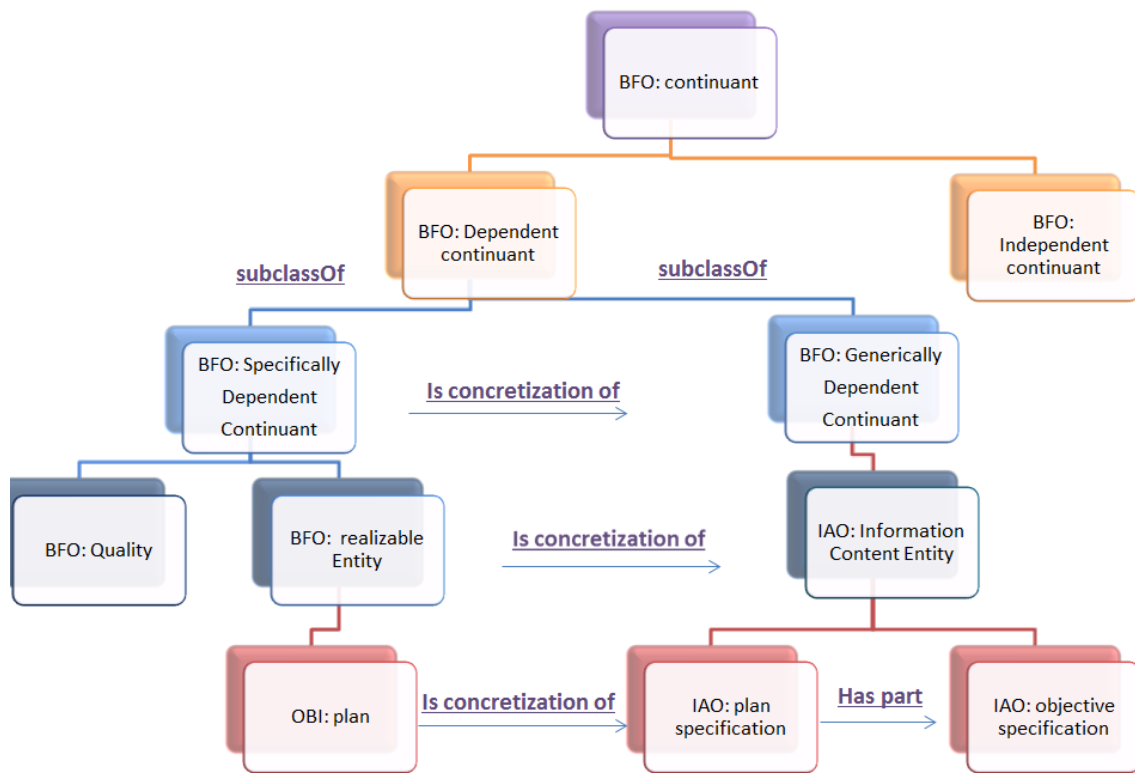


Figure 3.1: The main structure of the BFO and IAO ontologies, which OntoDM is extending

- datatype
- data mining task
- generalizations
- data mining algorithm

In figure 3.2 we can see the OntoDM extension of IAO, BFO, and OBI ontologies. The Algorithm Implementation class is defined as an OBI:plan descendant. OntoDM, also extends IAO:plan specification with Data Mining Algorithm class which represents an algorithm aiming to solve a data mining task. As a result the OntoDM mentioned classes are connected through the isConcretizationOf relation. Furthermore, a Data Mining Task is to produce a generalization from a given set of data and is defined as a subclassOf IAO:objective specification. Thus, Data Mining Algorithm has_part a Data Mining Task. A Data Mining Task also has_part an output data specification entity class, which represents the predictions (output) of an algorithm. A simple example of the above relations is a rbf kernel support vector machine (r_svm), which is concretization of a binary classification algorithm. A binary classification algorithm intends to solve (has_part) a binary

classification task, which produces (has_part) a binary prediction output. Finally, an Algorithm Implementation has_quality an OntoDM:parameter, which represents the tuning hyperparameters of an algorithm and is an extension of the BFO:quality class. Back, to the previous example, an r_svm has_quality the hyperparameters, cost, sigma e.t.c.

The above is the ontology structure of the main entities which describe a simple DM process and also the one that we exploited and extended for the purposes of our implementation.

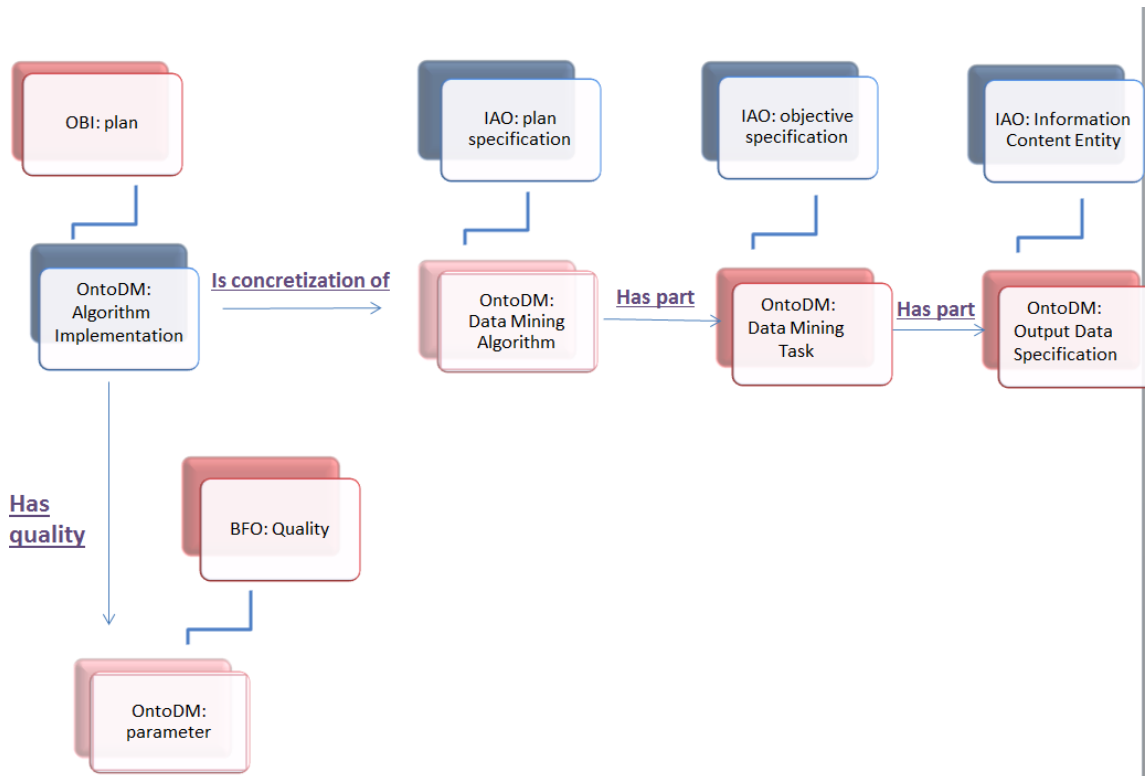


Figure 3.2: The structure of the OntoDM entities mainly used in RB-DMA. OntoDM:output data specification represents the target type of a dataset, OntoDM: Algorithm implementation represents an algorithm implementing a data mining algorithm, OntoDM: parameter represents the hyperparameters of a data mining algorithm implementation. The bold lines represent subClassOf relations, showing which are the OBI, BFO, and IAQ extended classes.

3.1.2 Incorporating and Extending OntoDM

The incorporation of a knowledge base describing the different data mining workflow phases is vital for the efficiency of our system. For this purpose we employed and partially extended the OntoDM ontology.

As previously mentioned, data mining task class represents the process which aims to produce a generalization from a given dataset. In data mining, there are 4 fundamental tasks which are also described in OntoDM, clustering, pattern discovery, probability dis-

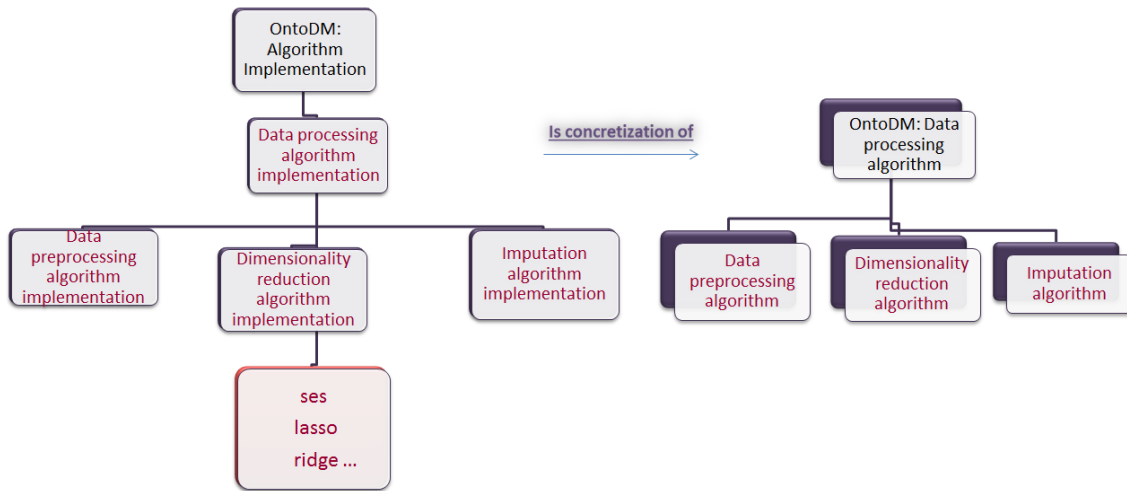


Figure 3.3: OntoDM extension, the red font rectangles represent the newly created entities. Data preprocessing algorithm, Dimensionality reduction algorithm, and Imputation algorithm are extending the OntoDM: Data processing algorithm, in order to represent the algorithms of the respective workflow phases. Imputation is part of the data preprocessing phase, but is represented as a separate entity. The entities represented as subclasses of the OntoDM: Algorithm Implementation entity are the algorithms implementing the respective workflow phases.

tribution estimation, and predictive modelling. The last one is further decomposed into primitive output prediction, and structure output prediction tasks. In our implementation we focused on the latter, which encapsulates binary, multi-class classification and regression tasks, which give output (has_part) boolean, discrete, and real respectively, defined in OntoDM as output data specification subclasses.

For the purposes of our system's implementation, we expressed our modelling phase algorithms as subclasses of the predictive modelling algorithm implementation entity, which itself extends the algorithm implementation entity. In order to also incorporate the rest of the DM workflow phases into the OntoDM ontology, we invented an entity, as it is shown in figure 3.3, which extends algorithm implementation, named data processing algorithm implementation, and has three descendants. Each one of them corresponds to a different workflow phase, data preprocessing algorithm implementation, dimensionality reduction algorithm implementation, and imputation algorithm implementation. The aforementioned classes are connected through the isConcretizationOf relation to the, also newly created, data preprocessing algorithm, dimensionality reduction algorithm, and imputation algorithm, respectively, which are all subclasses of the algorithm entity. Our system's in-charge methods are listed as subclasses of the corresponding implementation entity.

Furthermore, the classes representing the tuning hyperparameters of the system's methods are inheriting the OntoDM:parameter class. Each one of a method's hyperparameters is expressed as instance of the respective parameter class. Those classes are connected through the has_quality relation to the corresponding algorithm implementation subclasses. As a working example for all the above we can refer to figure 3.4 which

depicts the `r_svm` example, mentioned in the previous section.

Relation	Explanation
ExplanationEasiness	interpretability
hasTrainTimeConsumption	train time complexity
hasPredictTimeConsumption	predict time complexity
hasAvgAccuracy	average generalization efficiency
has_data_items_number	maximum sample size
has_features_number	maximum feature size

Table 3.1: The relations used to represent the data mining entity qualities in the explanation column.

Finally, we also introduced 4 relations and employed 2 already existing in OntoDM, in order to express some quality attributes for each algorithm. They all take a range (an integer between 1 and 5), which denotes the score related to each of the qualities for the respective algorithm. We present them in table 3.1.

3.2 The systems architecture

RB-DMA accepts as input the dataset to be analyzed, and executes a series of steps in order to output a suggested ranked list of compatible, for this analysis, workflow instances. The architecture of the system is depicted in figure 3.5. The first and last step, (**Requested task identification**, and **Final Workflow Composition and Ranking**), are responsible for processing the input and generating the final output, respectively. The intermediate ones correspond to the processes of identifying the appropriate methods for each of the workflow phases. Our approach mainly consists of two essential and complementary mechanisms. The first one is the knowledge base which is represented through the ontology described in section 3.1. SPARQL query language is used as an inference engine on the ontology, in order to retrieve the information required, throughout the different workflow instance constructing steps. The other one, is the rules mechanism, which consists of a set of heuristics, described in section 3.3, expressed using the rules management system, Drools. Its responsibility is filtering and processing the retrieved information. The system's steps are described in detail in the following paragraphs. For convenience and simplicity the bank dataset example is being used.

3.2.0.0.1 Requested task identification Our system, will read the bank dataset, and try to identify what is the requested data mining task. Given the binary target, and after inferencing on the ontology it concludes to a binary classification task. The blue arrow labeled as `DM_task` in fig 3.5 represents the data mining task, returned as output of the knowledge base, which is then used as input to the following steps.

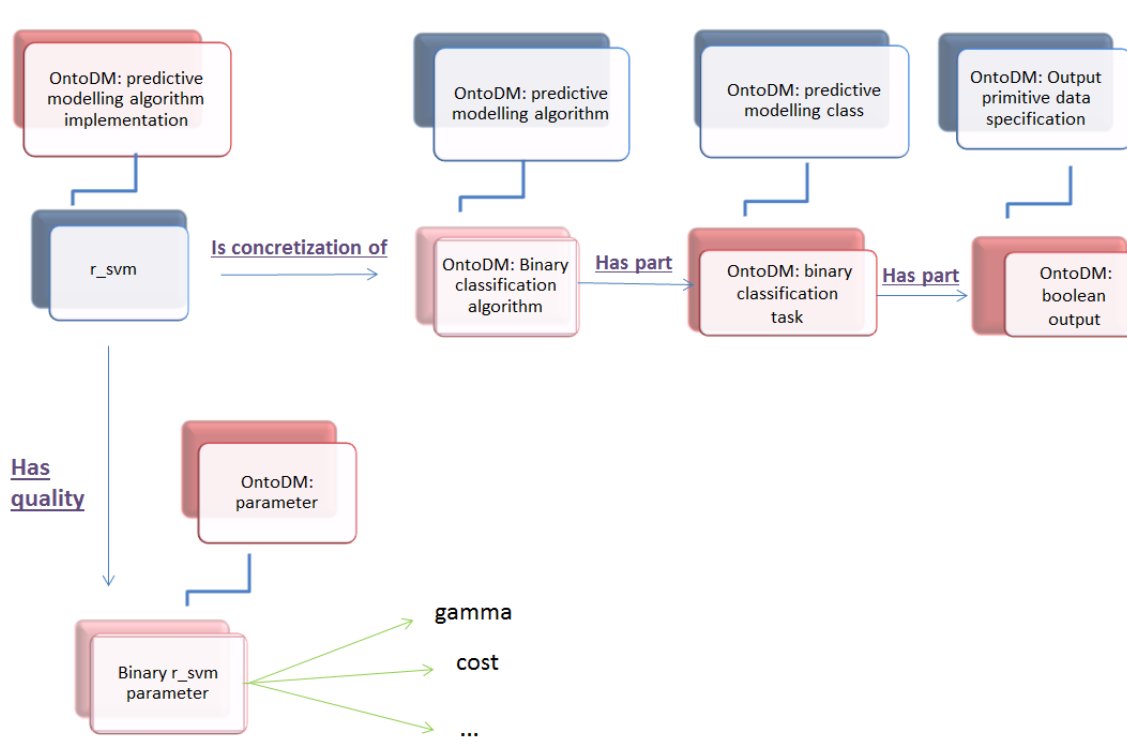


Figure 3.4: rbf kernel support vector machine (`r_svm`) use case example: Assuming a dataset with binary target (represented by `OntoDM: boolean output`), the system following the `Has part` relation, identifies a binary classification task (represented by `OntoDM: binary classification task`). Starting from the binary classification task entity, and following the `Has part` relation, it concludes that binary classification algorithms are the appropriate modeling components. Through the `isConcretizationOf` relation, the rbf kernel svm is returned as an implementation of a binary classification algorithm. Finally, the `Has quality` relation is being used, in order to identify the methods' appropriate tuning hyperparameters, `gamma` and `cost`, which are represented as `binary_r_svm parameter` instances.

3.2.0.0.2 Preprocessing Right after, there is a need to investigate if the dataset will be needing other preprocessing (imputation, standardization, feature selection) before trying to build a model describing it. In order to proceed, the system specifies the predictors datatype, verifies their completeness (possibility of missing values), and refers to the rules repository (later described in section 3.3.2) to identify the appropriate preprocessing components of the workflow instances. The dataset in our case has no missing values so there is no need to add imputation in the generated workflow instances. Most of the features are categorical, so RB-DMA decides to apply binarization, as *Long and Jeremy* suggest in [17], as first phase of all the generated workflow instances. The rest of the features are numerical (integers or doubles). It is a good practice in data mining to standardize the numerical attributes, ensuring that all values reside in a similar numerical range. Normalization, would be another valid option, but as it is claimed in [18], there is no significant difference between the impacts of the two methods. Because of that we decided to only include the former in our system, and thus it is added as next phase in the workflow instances. Finally, in this step a boolean parameter is set to a value indicating whether the

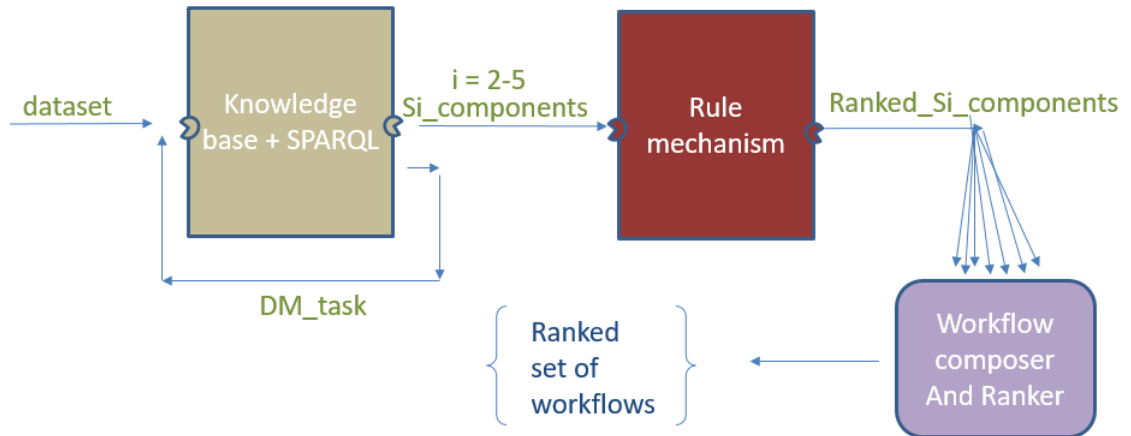


Figure 3.5: The system contains two main mechanisms, the knowledge base, which consists of data mining entities represented in the OntoDM ontology, and the rule mechanism, a set of rules expressed using the Drools system. The system executes 6 steps in order to compose and rank the proposed workflow instances.

The process of the first step is represented through the arrow denoting the dataset given as input to the knowledge base, and the DM_task arrow, which represents the identification of the appropriate data mining task, after reading the dataset. The arrow connecting the knowledge base to the rule mechanism, represents the possible components of the workflow instances for each of the four workflow phases (i). The arrows labeled as ranked_Si_components, represent the ranked and filtered workflow components, according to the rules of the rule mechanism. In the last step the workflow composer and ranker takes as input all the possible methods for each workflow phase, retrieved during the previous 4 steps, composes the appropriate workflow instances, and finally ranks them.

next phase will be feature selection or not. In our case the variable is set to true because the feature dimensionality is considered large enough to require reduction.

3.2.0.0.3 Feature Selection Considering rules related to the predictors amount (features), RB-DMA decides whether there is a need for adding feature selection in the workflow instances. Feature selection refers to the process of reducing the inputs for processing and analysis, or of extracting useful information or features from existing data. This phase is crucial before applying almost every machine learning algorithm, because it, (a) enhances the integrity of estimated efficiency, (b) reduces the model training time, (c) positively affects the final model's interpretability. For the current analysis, the system decides that feature selection would be beneficial. A query is performed on the ontology to obtain a set with the applicable methods for the current task. Elastic-net [19], is one of the options. It is a modeling method which is commonly used for the current task, because of its benefit to inherently give importance weights to the predictors, as part of the building process. Ses [20], is a multiple signature method which is also performing feature selection, and is the other member of the proposed set. Both are scalable methods and are proved to accurately reduce the features to the most important subset. Our sys-

tem generates appropriately ranked workflow instances having both of these methods as options, for the feature selection phase.

3.2.0.0.4 Modeling algorithm The next, and one of the most important workflow phases is modeling. At this point the system has to select the machine learning algorithms which are compatible to perform the machine learning task obtained in the first step. This is going to be accomplished again through inference. For the binary classification task in our example, all the available binary classification algorithms are going to be retrieved. RB-DMA employs Random Forests, support vector machine with linear, radial, and polynomial kernel, knn, Classification Decision Trees, and elastic-net.

3.2.0.0.5 Hyperparameter tuning An integral part of the successful workflow instance composition is the appropriate algorithm hyperparameter tuning of, both the feature selection and modeling steps. Hyperparameters are algorithm parameters whose values are set prior to the learning process. Their tuning process is crucial for each of the retrieved algorithms in order to succeed an optimized performance. To accomplish this, the system combines the power of its two main components, the knowledge base and the rules mechanism. The best performing ranges of hyperparameter values are obtained through the knowledge base, and afterwards efficient combinations of them are constructed under the guidance of the rules mechanism.

3.2.0.0.6 Model Selection and Performance Estimation The last two components, with major importance, are the proposed model selector and performance estimator techniques. The former is the method that is used to estimate the generalization performance of the models and to select the best. The latter, is the method used to estimate the real performance of the selected model. The most commonly used model selection and performance estimation methods, are “N fold Cross Validation (CV)” [21], “Hold Out (HO)”, and variations of them. During the execution of a Cross Validation process, the dataset is split into N equal sized folds. Recursively, N times, the amount of N -1 folds is used for fitting a model, whereas the last fold is kept for assessing the predictive ability of the model. Cross-validation, selects the model with the best average predictive ability, calculated based on all different fold splits (N in total). Hold Out is a simpler process, during which, the data is partitioned into 2 mutually exclusive subsets which are called a training set and a test set, or holdout set.

The main factor the system is taking into account in order to select the most appropriate method to apply is the dataset itself, the feature size, sample size, and the ratio between them. In the case of large sample size, with respect to the feature size, even a simple “hold out” model selector, and a “test set” as performance estimator, are enough. The rule mechanism is responsible for taking these decisions. In our working example the sample size is classified as large and the feature size as small. Consequently, the system proposes “3 fold cross validation” as a model selector and a simple “hold out” method for the performance estimation.

3.2.0.0.7 Final Workflow Composition and Ranking After executing the above steps, the appropriate workflow components are available, and the system needs to combine and rank them. The finally recommended workflow instances consist of all the possible combinations of the previously obtained components, minus some special cases. Despite what discussed in the **Feature Selection** step, some algorithms perform feature selection as part of the model building process, offering the benefit of eliminating the added overhead. Elastic-net is classified in this category and thus workflow instances (with or without feature selection) are produced when the modeling is processed by the algorithm. Furthermore, there is no workflow instance where both the feature selection and the modeling phases are processed by an elastic-net model. The compatibility, among the workflow instance's components, and the dataset, is assured due to the **Preprocessing** step. Finally, the system provides the user with a ranked list of the workflow instances, based on a weighted function of each components rank (described in section 3.3.1), which is obtained through both the systems mechanisms.

3.3 Scoring workflows using heuristics

3.3.1 The scoring function

The final objective of the system, as already discussed, is to provide the user with a ranked list of DM workflow instances, depending on his initial task, dataset, and demands. In order to accomplish this, a weighted function of three components (function 3.1) is employed to score each of the workflow phases.

$$R = w_a A + w_{tt} TT + w_{tp} TP + w_c C \quad (3.1)$$

$$A, TP, TT, C \in [1, 5] \quad (3.2)$$

$$w_a + w_{tt} + w_{tp} + w_c = 1 \quad (3.3)$$

- *A stands for Accuracy*
- *TT stands for Train Time Complexity*
- *TP stands for Predict Time Complexity*
- *C stands for Comprehensibility*
- *w_a, w_{tt}, w_{tp}, w_c , represent the "weights" for the respective function components*

The scoring function consists of A, TT, TP, and C, multiplied by the respective weight w , which is supplied by the user. Components with larger weight factor are considered of higher importance for his task. Summing up the *weights* should equal to 1 (equation 3.3). Accuracy, Train (Predict) Time Complexity, and Comprehensibility, as shown above

in equation 3.2, are taking values in the range [1,5], denoting the lowest and the highest score, respectively.

In order to obtain the final rank of a workflow instance we compose the scores of the separate phases using the relation 3.4

$$TotalR = \sum_{n=1}^{n=5} w_n * R_n \quad (3.4)$$

A workflow instance's final rank (TotalR) is a linear combination of its component's ranks, calculated using equation 3.1.

3.3.2 The scoring heuristics

Each workflow instance component is separately scored based on the rating function, described in section 3.3.1. Ontology reasoning combined with a set of heuristics are used to obtain each phase's matching algorithm. Furthermore, the same mechanisms are employed to calculate the values for the components of the respective rating function. The heuristics are referring not only on the methods quality and applicability, but also on the range and quality of the current tuning hyperparameters. The rest of this section describes and justifies the system's heuristics.

3.3.2.1 support vector machine performance heuristics

support vector machine	
Condition	Consequence
cost ↑	rbf.AvgAccuracy ↑
epsilon ↑	rbf.AvgAccuracy ↓

Table 3.2: Table of general support vector machine heuristics. Cost is an svm tuning hyperparameter, representing the svm penalty factor. Epsilon is another svm hyperparameter, used mainly when training regression models, affecting the support vectors quantity. AvgAccuracy is a metric, defined in our system, which represents the average performance capability of an algorithm, taking values in the range of 1 - 5, where 1 is the lowest performance capability, while 5 is the highest one

Linear Kernel support vector machine	
Condition	Consequence
FeatureSize > SampleSize	linear.AvgAccuracy = 4
FeatureSize < SampleSize	linear.AvgAccuracy = 2

Table 3.3: Table of linear kernel svm heuristics. AvgAccuracy \in [1 - 5].

A Support vector machine [22] is an algorithm which can be applied in both classification and regression tasks. According to [23], svm is one of the top performing methods compared to 169 others, on 121 UCI datasets. AvgAccuracy is a quality metric, defined in

rbf Kernel support vector machine	
Condition	Consequence
FeatureSize > SampleSize	rbf.AvgAccuracy = 2
FeatureSize < SampleSize	rbf.AvgAccuracy = 4
c == median(range)	rbf.AvgAccuracy = 5
-	gamma = 1 / cost

Table 3.4: Table of rbf kernel svm heuristics. The condition $c == \text{median}(\text{range})$ denotes, cost equal to the median of cost range. Gamma is a tuning hyperparameter for rbf and polynomial kernels, controlling the area of the support vectors influence. AvgAccuracy $\in [1 - 5]$.

polynomial Kernel support vector machine	
Condition	Consequence
FeatureSize > SampleSize	polynomial.AvgAccuracy = 3
FeatureSize < SampleSize	polynomial.AvgAccuracy = 3
-	gamma = 1 / cost

Table 3.5: Gamma is a tuning hyperparameter for rbf and polynomial kernels, controlling the area of the support vectors influence. AvgAccuracy $\in [1 - 5]$.

our system, which represents the average performance capability of an algorithm, taking values in the range of 1 - 5, where 1 is the lowest performance capability, while 5 is the highest one. We decided to use this higher range of values, instead of 1 - 3 (bad , medium, best), in order to better differentiate algorithms with slighter differences on their qualities. In [23] they concluded that $\text{rbf svm AvgAccuracy} > \text{polynomial svm AvgAccuracy} > \text{linear svm AvgAccuracy}$. The previous relation is valid, when the sample size of the data set is larger than the feature size. When this is not the case (Feature size > Sample size) , the dataset is possibly linear and, consequently linear methods are more promising. Thus, the opposite relation, $\text{rbf svm AvgAccuracy} < \text{polynomial svm AvgAccuracy} < \text{linear svm AvgAccuracy}$, is the valid one, in this case. These heuristics are expressed in rows 1 - 2 of tables 3.3 - 3.5.

Hyperparameter tuning is viable for a method. By using it, one can tremendously affect the methods complexity and performance. A good choice of hyperparameter values can lead to extremely accurate results while in the opposite it may lead to extremely inaccurate results. For this reason, our system ranks differently the multiple configurations of the same method. Table 3.2 lists the support vector machine hyperparameter rankings.

Svm hyperparameter cost represents the penalty factor. Quoting *Alpaydin et Ethen* in [24] "If it is too large, we have a high penalty for nonseparable points and we may store many support vectors and overfit. If it is too small, we may have underfitting." In other words, higher values for cost almost exactly describe the training set, leading to high training accuracy, but may not operate best on test set. Lower values, describe the training set in a general(abstract , constrained) way, resulting in poor performances.

Based on the above, according to scikit-library documentation⁵, best practise is choosing cost values in a logarithmic grid of 10^{-3} to 10^3 . In our approach we are using values in the grid 10^{-1} to 10^2 , and scoring them in ascending order (table 3.2, first cell). While testing those rules on multiple datasets, we noticed that on the svm with an rbf kernel, the middle value of the previous cost grid (i.e cost = 10) presents way better performances than the rest of the grid values, a fact that verifies the above mentioned theory. Consequently, we adopted the heuristic located in the third cell of table 3.4, giving higher priority between rbf support vector machine methods to those having cost equal to 10.

Gamma is a parameter which controls the area of the support vectors influence. If gamma is too high, the radius of the area only includes the support vector itself and so the model overfits. When gamma is very low, the radius of the area is almost containing the whole training set, resulting again in poor dataset description. Concluding from the above, gamma and cost are inversely proportional on influencing the method's accuracy. According to some experiments reported in the scikit-library documentation, the results of which are depicted in figure 3.6, the best performing svm configurations consist of cost and gamma values which are connected through the relation depicted in tables 3.4 and 3.5.

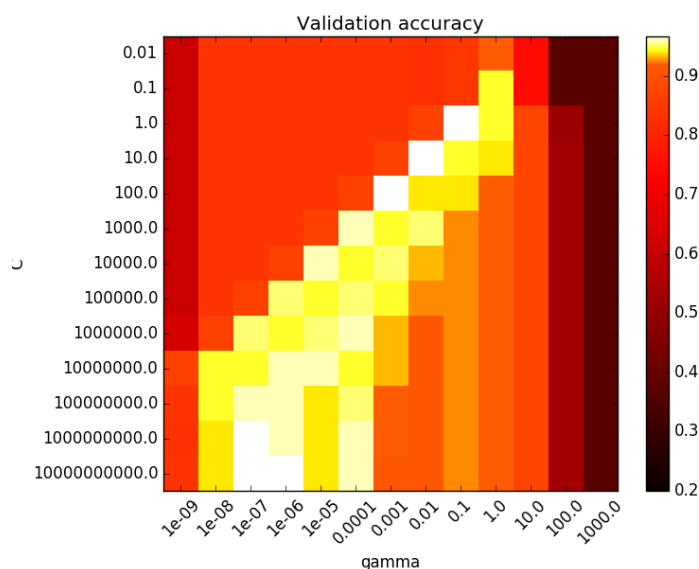


Figure 3.6: Best performing cost and gamma configurations, according to scikit-library documentation's experiments.

Another support vector machine hyperparameter, which is only used on regression tasks, is epsilon. Its role is to control the width of the epsilon-insensitive zone, used to fit the training data. The value of epsilon affects the support vectors quantity. The higher the epsilon value is, the fewer the support vectors will be. Hence, epsilon affects the model's effectiveness and complexity, like cost does, but in an opposite way (table 3.2, second cell).

⁵scikit: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

3.3.2.2 elastic-net performance heuristics

elastic net (modeling)	
Condition	Consequence
FeatureSize < SampleSize	en.AvgAccuracy = 3
SampleSize < FeatureSize	en.AvgAccuracy = 4
FeatureSize == vsmall	en.AvgAccuracy = 2
Alpha ↑	en.AvgAccuracy ↓
Lambda ↑	en.AvgAccuracy ↓

Table 3.6: Table of elastic-net heuristics. Alpha is the elastic-net mixing hyperparameter, controlling the weighted combination of L1 and L2 penalties of the model. Lambda is the hyperparameter controlling the amount of feature coefficients shrinkage. AvgAccuracy $\in [1 - 5]$.

Elastic net [25] is a regression method, compatible with classification and regression tasks, that linearly combines the L1 and L2 penalties of the lasso [19] and ridge[26] methods.

Referring to the experiments of *Fernandez et.al* in [23], we conclude that elastic-net, as classifiers, are medium - performing methods, and thus our system ranks them with a value of 3, see No 1 heuristic of Table 3.6. Elastic-net is mainly efficient when FeatureSize > SampleSize. After conducting experiments, we ended up that in this case the method achieves better performances (heuristic No 2 Table 3.6). Finally, we noticed that in very small dimensionality datasets the performances seem to be poor (heuristic No 3 Table 3.6).

In case of the lasso performing as a feature selection method, we decided to rank its AvgAccuracy, which, in this case stands for "how descriptive are the selected features", with a value of 4. We concluded to this, due to the fact that on average lasso accurately selects the most descriptive features, increasing the predictive performance of the most modeling methods. Alpha is the Elastic net's mixing parameter. It takes values in the range of (0,1]. The upper and lower values represent lasso with L1 penalty and ridge with L2 penalty, respectively. Models with intermediate Alpha values are weighted combinations of the two methods. *Waldmann et.al* in [27] conducted some experiments comparing lasso and multiple elastic-net configurations on real and simulated genome-wide data. The results point out that configurations with lower values of Alpha achieve better performances and more accurately select the important features, compared to those with higher values. According to these results we incorporated to our system the heuristic No 4 in Table 3.6. Finally, Lambda is elastic-nets shrinkage parameter. When Lambda equals to zero, no coefficient shrinkage is performed. As Lambda increases the feature coefficients are shrunk even more strongly, resulting to even less features selected as important. Deciding the amount of features that need to be selected (more or less), most of the times depends on the problem. Selecting a very small Lambda may cause misleading results, due to irrelevant features being part of the final model. On the other side, setting Lambda with very large values may lead to the elimination of high relevance features, resulting to underestimating performances. Thus, in our implementation the main system performs a five-fold

cross validation of the elastic-net in order to select the best performing Lambdas. These values are afterwards ranked in an ascending order, meaning that as Lambda increases the models AvgAccuracy decreases (heuristic No 5 in Table). This is decided because most of the times the highest selected Lambda value results to a feature set of size one or two, and thus to seriously underestimated performances.

3.3.2.3 Random Forest performance heuristics

Random Forests	
Condition	Consequence
Always	rf.AvgAccuracy = 4
NumPredictorsToSample \uparrow	rf.AvgAccuracy \downarrow
minleaf \uparrow	rf.AvgAccuracy \uparrow
NumPredictorsToSample	$x * \sqrt{FeatNum}$

Table 3.7: Table of Random Forests heuristics. NumPredictorsToSample is the number of features to select at random for each decision split, for a decision tree of a random forest. Minleaf, controls the minimum accepted size for the leafs of a decision tree. FeatNum denotes the dataset dimensionality. AvgAccuracy $\in [1 - 5]$.

As already discussed, radial support vector machines is the best performing classification algorithm. Random Forests [28] [29] is a classification and regression method, which, according to the aforementioned comparison study [23], performs competitively well. Thus, our system is equally rating their AvgAccuracy, as it is shown in table 3.7 heuristic 1. Referring also to other studies comparing regression methods on datasets of manifold fields, such as student performance [30], and spatial data [31], we are verifying this conclusion. The only exception is in [32], which also refers to spatial data, where random forests seem to perform worse than svm.

NumPredictorsToSample is the number of features to select at random for each decision split, for a decision tree of a random forest,. According to *Bernard et.al* in [33], NumPredictorsToSample = $\sqrt{FeatNum}$ is one of the best performing value settings. In our approach, we are proposing multipliers of this setting, as it is shown in heuristic No 4 of Table 3.7. *Bernard et.al*, after conducting experiments on a number of datasets, concludes that, in most of the cases, best performances are succeeded for smaller values of the hyperparameter (heuristic No 2).

Finally, minleaf is another crucial hyperparameter for the Random Forests method. The larger leaf size, the smaller tree will be build. In general building smaller trees is desirable, because they are less biased to the training set, and so overfitting is prevented. Based on this, our system employs the heuristic No 3.

3.3.2.4 knn and Decision Tree performance heuristics

Knn and Decision Trees are not proven to be in the top performing algorithms [23], and thus our system lower ranks them (heuristic No 1 Table 3.8 and 3.9). Knn is a classification

knn	
Condition	Consequence
Always	knn.AvgAccuracy = 2
k ↑	knn.AvgAccuracy ↓

Table 3.8: Table of knn heuristics. K is the hyperparameter controlling the amount of neighbors considered in the knn model. AvgAccuracy \in [1 - 5].

algorithm, and Decision Tree is capable of performing both classification and regression tasks.

K is knn's tuning hyperparameter, controlling the number of neighboring instances that the algorithm considers in order to predict the target value of a new point. The less neighbors the algorithm considers the better the prediction, and thus the system introduces heuristic No 2 in Table 3.8.

Decision Tree	
Condition	Consequence
Always	dt.AvgAccuracy = 2

Table 3.9: Table of Decision Trees heuristics. AvgAccuracy \in [1 - 5].

3.3.2.5 ses performance heuristics

ses	
Condition	Consequence
Always	ses.AvgAccuracy = 4
maxk ↑	rf.AvgAccuracy ↑
threshold ↑	rf.AvgAccuracy ↓

Table 3.10: Table of ses heuristics. Maxk sets the maximum conditioning set to use in the multiple conditional independence tests performed. Threshold is the parameter used for assessing the p-value significance in the independence tests. AvgAccuracy \in [1 - 5].

Ses [20] is a feature selection method. Both Ses and elastic-net feature selection algorithms, are scalable, and efficiently reduce the features to the most important subset. Due to this fact we have equally ranked their AvgAccuracy, as it is illustrated in heuristic No 1 of Table 3.10. The efficiency of this method is tightly connected with the setting of two hyperparameters, maxk and threshold. In order to perform feature selection, ses is conducting multiple conditional independence tests, maxk sets the maximum conditioning set to use in them. The larger the set size, the more accurate the selected feature set (heuristic No 2 Table 3.10). Threshold is the parameter used for assessing the p-value significance in the independence tests. Two values are considered significant when the

pvalue is less than the threshold. As it is easily understood for a smaller pvalue the features selected are more descriptive, and thus our system employs the heuristic No 3.

3.3.2.6 Explanation easiness (interpretability) heuristics

Method	Interpretability
SVM	2
Random Forests	2
Decision Tree	4
knn	3
elastic-net	5

Table 3.11: Rank of interpretability for the 5 modeling methods

Decision Tree is a logic-based algorithm and thus it is easily explained [34]. The random forests consist of many decision trees, but considering that they can be quite numerous, and that the features selected for splitting are different subsets, random forests, are not easily interpretable compared to the decision trees. Elastic - net actually is a simple linear regression model, so it is also easy to explain and interpret it. Furthermore, knn is considered to have very poor interpretability because it consists of nothing more than an unstructured collection of training instances. Finally, SVM produce the most complex and uninterpretable models. Table 3.11 depicts the ranks assigned by our system for the interpretability of the methods.

3.3.2.7 Train and Predict complexity heuristics

Complexity		
Method	Train	Predict
SVM	S^3	$S * F$
Random Forests	$T * F * S * \log(S)$	S
Decision Tree	$F * S * \log(S)$	S
knn	$F * S$	$k * S$
elastic-net	$\min(F, S) * F * S$	$\min(F, S)$
ses	$F * \min(F, S)^{maxk+1}$	$\min(F, S)$

Table 3.12: Train and Predict Complexity of the methods (used as ranking heuristic). S represents the sample size, F represents the feature size, k , T , and $maxk$ are the selected hyperparameter values for the respective algorithms.

Ranking data mining algorithms based on their train and predict time requirements is quite complex. This is due to the fact that there is not a global value for these measurements. They depend on the algorithm implementation that is being used, but also on the machine used to run the algorithm. To perform a fair and decent scoring we ended

up leveraging the (train and predict) complexity of the algorithms as scoring functions, presented in Table 3.12.

In order to achieve this, we used the complexities as functions of the sample and feature size of each dataset. The values returned are normalized in the range of [1-5] as the rest of the rankings.

3.3.2.8 Preprocessing heuristics

Preprocessing	
Condition	Consequence
Missing values	Imputation
Categorical Features	Binarization
Numerical Features	Standardization
NumFeat > 10	Feature Selection

Table 3.13: Preprocessing Stage heuristics. NumFeat represents the dataset dimensionality.

RB-DMA decides to binarize the categorical features, when they exist, as *Long and Jeremy* suggest in [17]. In case the dataset contains numerical features (integers or doubles), it is a good practice in data mining to standardize them, ensuring that all values reside in a similar numerical range. Discretization, would be another valid option, but as it is claimed in [18], there is no significant difference between the impacts of the two methods.

Finally, more than 10 features in the dataset will possibly contain irrelevant, or unwanted information, and thus the system decides to add feature selection as part of the workflow instances to keep only the useful information.

3.3.2.9 Model Selection and Performance Estimation heuristics

The accurate selection of the appropriate model selector and performance estimator methods is essential to produce a reliable result. Picking the wrong method for these workflow phases may lead to overestimating the algorithm performance, and/or selecting the wrong one as best performing. Table 3.14 presents the heuristics guiding the selection of the appropriate methods, sorted in the order that the system evaluates them. When a true condition is reached the algorithm stops and returns the resulted methods. For instance, if both the first and the third conditions are true, then the first is the appropriate one.

The most important factor is the data size. We have classified the sample and feature sizes into 5 categories, presented in Table 3.15, and 3.16, respectively. In the case where the sample size is very large, there is no need to spend time on training the models multiple times. It is equally effective to split the dataset in 3 parts. The train set, for training the models, the test set, needed to select the best model, and the validation set, in order to assess the true performance of the selected model. This is what heuristic No 1 demonstrates. For a large sample size, we should use a 3 fold cross validation for the model selector and a hold out validation set as the performance estimator. If the dataset

Condition	Model Selector	Performance Estimator
VLarge (<i>SampleSize</i>)	HO ($TS = 0.3 * TrainSize$)	HO ($VS = 0.3 * SampleSize$)
Large <i>SampleSize</i>	3 FOLD CV	HO ($VS = 0.3 * SampleSize$)
VLarge <i>FeatSize</i>	3 FOLD CV	BBC-CV Method
Diff = 2 or Large <i>FeatSize</i>	5 FOLD CV	BBC-CV Method
Diff = 1	10 FOLD CV	BBC-CV Method
Diff = 0	5 Repeats 3 FOLD CV	BBC-CV Method
Diff = -1	5 Repeats 5 FOLD CV	BBC-CV Method
Diff = -2	3 Repeats 5 FOLD CV	BBC-CV Method
Task = classification and Folds > min(ClassSize)	FOLD = min(ClassSize)	

Table 3.14: The Model Selector and Performance Estimator stage heuristics. The evaluation of the heuristics conditions is executed sequentially. The process stops when a condition evaluates to true. Diff equals $|SampleRank - FeatureRank|$, for example small - vsmall = 1. min(ClassSize) represents the size of the dataset's smallest class. TS stands for Train Set, and VS stands for Validation Set.

Sample	
Size	Rank
1 - 99	vsmall
100 - 999	small
1000 - 9999	medium
10000 - 49999	Large
> = 50000	VLarge

Table 3.15: Sample size ranks

Feature	
Size	Rank
1 - 9	vsmall
10 - 99	small
100 - 999	medium
1000 - 9999	Large
> = 10000	VLarge

Table 3.16: Feature size ranks

dimensionality is larger than medium sized, the feature selection process is time consuming. In order to avoid executing it multiple times when we have more folds, as it is shown in heuristic No 3, and No 4, we choose for model selection, 3 fold cross validation and 5 fold cross validation, respectively. In order to evaluate the third heuristic of the table, we have already rejected the first two. This denotes that the dataset should be at most medium sized. Due to this lack of data observations a simple "Hold Out" performance estimator is not enough. We choose to apply the Bootstrap Bias Corrected CV (BBC-CV) method as the performance estimator, which is equally well performing as the Nested Cross Vali-

dation (NCV) method [35], while it is less time consuming than that. In heuristics No 4 - 8 Diff represents the difference between the sample size rank and the feature size rank. For example, Medium SampleRank - Large FeatureRank = -1. The smaller the Diff, the larger the amount of folds. A good practice in this case, is also to perform repeated cross validation, as proposed in heuristics No 6 - 8. In the case where diff equals -2 the feature size is medium size rank, and thus, for the same reason explained earlier, we perform less repeats compared to the previous heuristics.

Finally, the last heuristic of Table 3.14, is the only one that is evaluated after the appropriate methods have been selected. It is a common heuristic in data mining that, for classification tasks, the number of folds should be no more than the size of the datasets' smallest class, when performing cross validation. Thus, if the previously selected amount of folds is greater, then, we set it equally sized to the smallest class. This is happening because, if more folds are selected for the cross validation, then, when splitting the dataset in a stratified way, one, or more of the folds will not have representative instances of the smallest class.

Chapter 4

Evaluation

4.1 The datasets analyzed

In this work we claimed that RB-DMA is a system capable of covering a wide range of 200 different data analysis scenarios, eliminating the prior separate interpretation for each one of them. Due to time limits, we were not able to test all of the scenarios. To support our claims, we managed to generate workflow instances for datasets from manifold fields, with different, sizes, descriptor data types, and preprocessing needs. The datasets are presented in Table 4.1. Five of them, indicated in column "B", were already analyzed by *Bernstein et.al*, in [4]. We chose to analyze them ourselves, in order to compare our results to the Bernstein et. al results, due to the resemblance of our works . The column named "NaN" indicates the datasets having missing values. The tested data mining tasks are binary classification and regression.

4.2 Evaluating the system

The first thing to be evaluated on an automated system, is the ability to complete its intended task without inconsistencies or missing results, and the ability to complete that, in a bearable amount of time.

For all the datasets that we gave as input to the system, there were no inconsistent or missing values in the answer set (produced workflow instances). This is due to the consistency and completeness of the ontology employed to act as the system's knowledge base. Furthermore, the user does not need to wait for a long period of time in order to obtain the requested results. It takes just a few seconds for RB-DMA to read the dataset, coordinate the cooperation of its main mechanisms and finally generate the appropriate workflow instances. Table 4.2 presents the amount of time needed to produce results for some of the (later described) datasets. In order to conduct a fair evaluation we recorded the execution time for multiple datasets, and multiple sample sizes. The second column of the table presents the time needed for the R `glmnet` function of the R CRAN package ¹, which is

¹glmnet: <https://cran.r-project.org/web/packages/glmnet/glmnet.pdf>

called through the rJava package ², to analyze the dataset and return the best operating lambda values for the elastic-net method. We decided to subtract this value from the total running time, and consider the values in the third column as the real execution time of the system.

The system's execution time is tightly connected and depended on the size of the dataset that is being processed. As it is shown in Table 4.2, p53_1000, dexter, and bank, which, regarding the dimensionality and/or the sample size, are large or very large datasets, so they require the highest execution time. CASP, which is also a large sample dataset seems to need less execution time, considering the fact that no feature selection queries are executed in this case, while the total time value needed was almost equal to the ones with feature selection.

The mean time required to produce the ranked workflow instances was 15 seconds, the lowest possible time we recorded was 4.6 seconds and the maximum went up to 20 seconds.

4.3 Evaluating the proposed workflow instances

In order to test our system's efficiency, we conducted a series of experiments. We collected 31 manifold datasets. The workflow instances, generated by the system, were executed using another automated system. The final results are demonstrated and evaluated in the following section.

4.3.1 Data analysis results

For each dataset, two methods were employed in order to evaluate the generated workflow instances. First, we calculated the Spearman's r_s , between the rank generated by our system and the true rank of the workflow instances after executing them on the current dataset. We did that, in order to assess the efficiency of our system's rankings, as well as to compare our results to those of *Bernstein et.al*, where the same evaluation method was used. To demonstrate our next, "first K workflows" claim, we created plots for each dataset indicating the maximum performance obtained after executing the first K of the generated workflow instances, where $K \in [1 - NumWorkflows]$.

In Tables 4.3, and 4.5 we depict the Spearman's correlations of the classification and regression datasets, respectively. The last 5 rows of table 4.3 also contain a second correlation, which is the one scored by *Bernstein et.al* for the same dataset.

Analysing the classification correlations of Table 4.3, we noticed that, almost 50% of them (9/20) have scored a r_s equal to 0.5 or above, and a 30% is close to 0.6 or more.

Table 4.4 presents the median and mean ranking correlations of the classification datasets, for both systems. The same metrics are also presented for the 5 datasets and also analysed by both systems. At first sight, the mean ranking correlations scored by the two systems, for both cases, are close to each other. In 2 of the 5 datasets RB-DMA outperforms IDEA. The opposite occurs 3 times, thus the two systems are almost equally outperforming each

²rJava: <https://cran.r-project.org/web/packages/rJava/rJava.pdf>

other. Comparing the scored median values for the two systems, gives the impression that IDEA is a more accurate one. It is noticed that the mean and median correlations scored by our RB-DMA are almost equal. This fact, empowers the certainty that our system is capable of generating equally accurate workflow instances for any classification dataset. At the same time, IDEA, as we observed, seems to produce much higher median than the mean value, which could indicate a less confident system.

Furthermore, our implementation integrates 3 more stages in the workflow, model selection, performance estimation, and hyperparameter tuning, in order to produce a fully featured system, which also justifies the resulted lower mean and median Pearson's values of our system.

Based on the above remarks, the two systems can be considered as competitive, at least for the binary classification task

The resulted correlations for the regression datasets are not equally satisfying . Regression, is a more difficult and complex analysis task. In this case, probably, more dataset characteristics, (such as feature variance and standard deviation, which are also used on meta-learning analysis tasks) need to be considered in the rules mechanism. However, 7 out of the 11 regression correlations (with value between 0 and 0.4), presented in Table 4.5, are better than a random execution order. Another 3 of them are close to, or above 0.5. In any case, there is still place for improvement regarding this analysis task.

In Figures 4.1 - 4.7 the results of our second evaluation method are presented. The plots represent the maximum performance (AUC for classification, and R^2 for regression) obtained, after executing the first K ranked of the proposed workflow instances for each dataset. These figures were generated using the following process

1. for N in $NumWorkflows$
 - (a) Run the first K workflow instances using the appropriate model selector
 - (b) Select the best performing one
 - (c) Estimate the performance of the selected model on a hold out test set
 - (d) Save the estimated as best performance
2. plot the set of best performances

We decided to use a test set as a performance estimator due to time limits. Any other method would require a prohibitive amount of time, in order to be executed N times.

Initially, the Figures are grouped based on the data mining task (classification Figure 4.2 - 4.4 , regression Figure 4.5 - 4.7). Next they are grouped based on the sample size (Figure 4.2 depicts classification datasets with very small and small sample size, Figure 4.3, medium sample size, Figure 4.4, large sample size, while Figure 4.5 depicts regression datasets with very small and small sample size, Figure 4.6, medium sample size, Figure 4.7, large sample size), and finally, based on missing values (datasets with a * on their name, are those having missing values). Figure 4.1 is separately presenting the *Bernstein* five datasets results.

There are cases where, the method chosen as best, during the model selection process, is

not the truly optimum one. In these cases, the performance reported during the performance estimation process may be lower than the expected. This phenomenon is visible in the figures below, by sudden drops of the line describing the performance.

Our main intention is to examine if the relation $K < 0.5 * NumWorkflows(1)$, (where $NumWorkflows$ is the amount of the proposed workflows) is satisfied for the majority of the cases. In other words, what we want to report is the point on the x axes where the line of the graph reaches its maximum value on the y axes. Starting with the results depicted in Figure 4.2, we can see that for all the datasets in this figure, K equals 1, strongly proofing our claim that at least for very small and small data sizes, executing only the first workflow instance will produce the best analysis. In the parkinson and Nomao_50 figures, a reduction of the performance is observed, in a portion of the first $0.1 * NumWorkflows$. Even if the user decides to execute K workflow instances, that are all contained in the "not the best performing" portion, the final performance will still remain very close (more than 90%) to best.

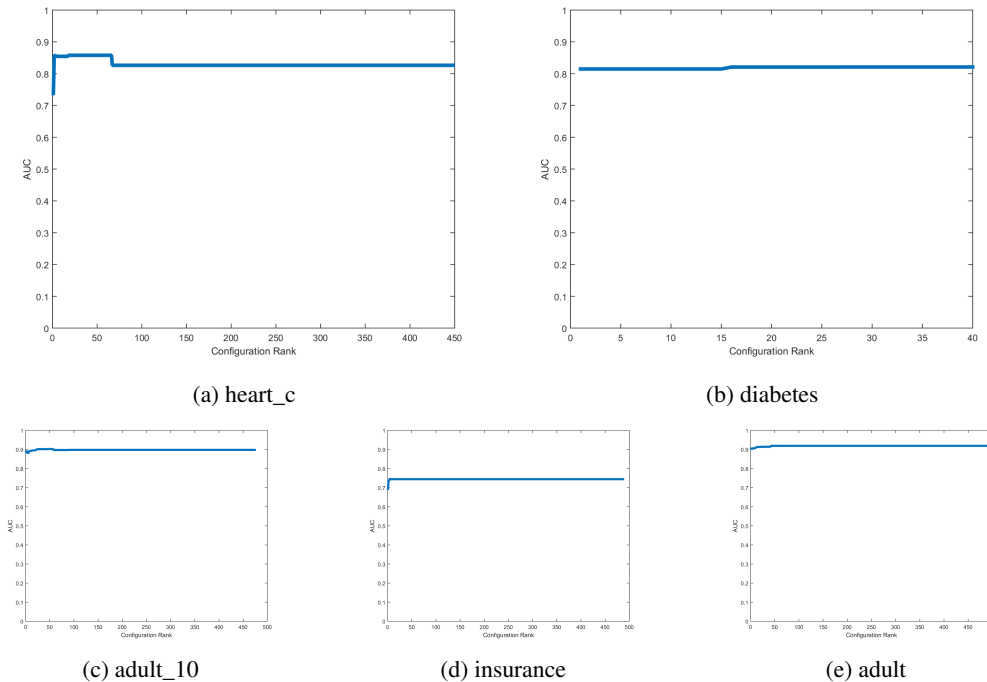


Figure 4.1: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Bernstein datasets.

Regarding the medium sized classification datasets of Figure 4.3, the results are almost equally satisfying, compared to those of the very small and small sized. The amount of worst case K is calculated around 5 or less, considering the secom, farmads, gina and hiva subfigures, resulting to the satisfaction of relation (1). ($K - 5 \ll 200$). Once more the best case K equals to 1.

Finally, the plots demonstrated in Figure 4.4 and 4.1, represent the performance of the

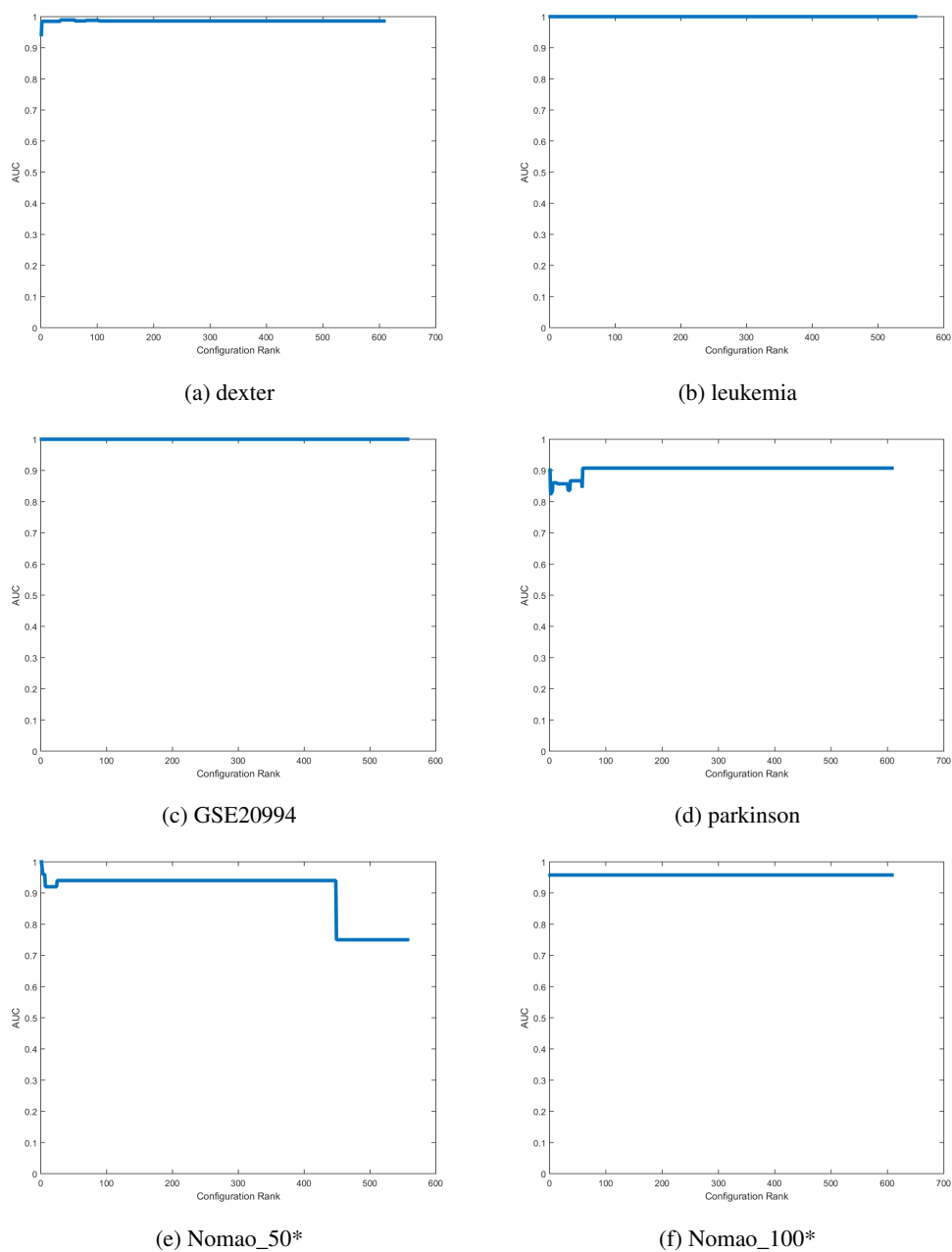


Figure 4.2: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. V_{small} , small sample sizes Classification, (* = dataset with missing values)

large sized datasets and the 5 *Bernstein* datasets, respectively, and confirm that relation (1) is satisfied for any kind of classification task, and most of the times for a K value equal

to one.

Regarding the regression datasets, the expectations should be lower, considering the results of their ranking correlations. Figure 4.5 illustrates the plots of the very small and small sized regression datasets. At a first sight, it appears that the phenomenon of the “misjudged as best selected methods” is more frequent. Not surprisingly, the value of K for the studentmat dataset is the smallest one and the only one dissatisfying relation (1). Even in this worst case, though, the violation of relation (1) depends on the user requests. After executing less than 20 workflow instances ($K < 20$), the system scores an R^2 equal to 0.1, which is the closest to best, compared to the rest of the performances in the plot. If the user is willing to sacrifice the best possible performance for the shake of the time saving, the system can be again considered as successful, and relation (1) could possibly still considered as satisfied ($20 < 340$).

Disregarding the previous result, the rest of the Figure 4.5 results are strengthening the validity of our claim, even for the regression task. The best K equals to 1, while in the worst case K equals to 20 out of more than 600 workflow instances.

Moving to the next rank size, Table 4.6 presents the results of the medium regression sample size. The values of K here are ranging from 1, remaining still the best case, to almost 500 (out of 600) that are dissatisfying relation (1).

Finally, there is only 1 dataset analyzed for the large sample size of the regression task. More results would be needed to extract strong conclusions. Based on the results, a value of 3 (out of 60) is obtained for K , scoring an R^2 equal to 0.6, very close to best, which is 0.62, a result which is enforcing our claim.

Concluding, we found out that for most of the cases, although the confidence was less and the amount of the workflow instances (K) were more, compared to classification, the regression tasks are still satisfying relation (1).

Name	Sample Size	Feature Size	Sample Rank	Feature Rank	Task	NaN	Descriptors	B
airfoilsnoise	1503	5	medium	vsmall	Regression		Continuous	
yacht	308	6	small	vsmall	Regression		Continuous	
diabetes	768	8	small	vsmall	classification		Continuous	✓
CASP	45730	10	large	vsmall	Regression		Continuous	
heart_c	294	13	small	small	classification	✓	Continuous	✓
adult_10	3256	14	medium	small	classification	✓	Mixed	✓
adult	32561	14	large	small	classification	✓	Mixed	✓
cadata_50	50	16	vsmall	small	Regression		Continuous	
cadata_500	500	16	small	small	Regression		Continuous	
cadata	5000	16	medium	small	Regression		Continuous	
bank	45210	17	large	small	Classification		Mixed	
parkinson	195	22	small	small	classification		Continuous	
studentmat	395	30	small	small	Regression		Mixed	
ada	4562	46	medium	small	Classification		Continuous	
insurance	9822	86	medium	small	classification		Mixed	✓
Nomao_50	50	120	vsmall	medium	classification	✓	Mixed	
Nomao_100	100	120	small	medium	classification	✓	Mixed	
Nomao_1000	1000	120	medium	medium	classification	✓	Mixed	
CommunitiesCrime	2215	125	medium	medium	Regression	✓	Mixed	
sylva	14394	213	large	medium	classification		Continuous	
blogData_50	50	281	vsmall	medium	Regression		Mixed	
blogData_100	100	281	small	medium	Regression		Mixed	
blogData_1000	1000	281	medium	medium	Regression		Mixed	
secom	1567	590	medium	medium	classification	✓	Continuous	
GSE20994	57	864	vsmall	medium	Classification		Continuous	
gina	3468	970	medium	medium	Classification		Continuous	
hiva	4229	1617	medium	large	Classification		Continuous	
leukemia	39	3051	vsmall	large	Classification		Continuous	
p53_1000	1000	5408	medium	large	Classification	✓	Continuous	
dexter	600	11035	small	vlarge	Classification		Continuous	
farmads	4143	54877	medium	vlarge	Classification		Continuous	

Table 4.1: Table presenting the datasets analyzed to evaluate the system. The datasets having underscore and a number, concatenated to their name are subsamples of the original ones. The last column indicates those that were also analyzed by *Bernstein et al* [4]. (vsmall and vlarge refers to very small and very large sizes, respectively).

Name	Total time	glm time	Real Time
insurance	45.25	40.1	5.15
heart_h	5.68	1.05	4.6
bank	21.5	1.88	20.0
Nomao_100	6.1	1.1	5.0
Nomao_1000	6.7	1.2	5.5
sylva	84.5	78.3	6.2
blogData_50	6.1	1.1	5.0
blogData_1000	8.6	3.6	5.0
CASP	8.1	3.0	5.1
dexter	27.7	8.4	19.3
p53_1000	20.0	5.2	14.8
mean			15.5
median			5.32

Table 4.2: Time needed to generate the workflow instances for 11 different datasets, evaluated in seconds

Name	r_s	Br_s
ada	0.50	
leukemia	0.41	
hiva	0.58	
GSE20994	0.44	
bank	0.67	
parkinson	0.48	
farmads	0.62	
sylva	0.24	
gina	0.54	
secom	0.43	
dexter	0.68	
p53_1000	0.08	
Nomao_50	0.39	
Nomao_100	0.35	
Nomao_1000	0.63	
heart_c	0.46	0.62
diabetes	- 0.08	- 0.52
adult	0.56	0.80
adult_10	0.48	0.75
insurance	0.64	0.31

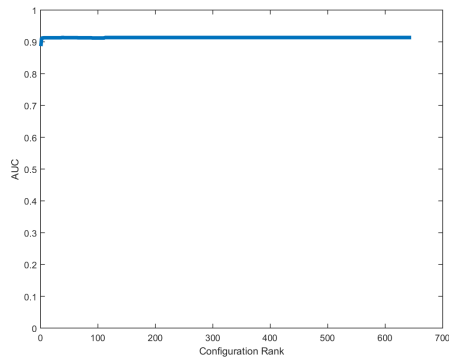
Table 4.3: Pearson's r_s between the system's workflow instance ranking, and the actual one, for the classification task datasets. Br_s column depicts the correlations resulted in *Bernstein et.al* [4] for the same datasets.

	mean	median
5 datasets analyzed by both systems		
IDEA	0.39	0.62
RB-DMA	0.41	0.49
Classification (general)		
IDEA	0.53	0.70
RB-DMA	0.46	0.48

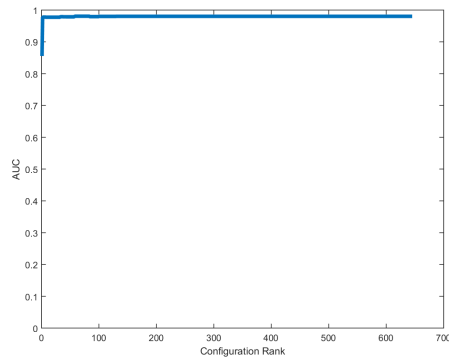
Table 4.4: Comparing IDEA and RB-DMA mean - median Pearson's r_s

Name	r_s
CASP	0.50
cadata	0.28
yacht	0.34
studentmat	0.01
CommunitiesCrime	0.24
cadata_50	0.09
cadata_500	-0,07
blogData_100	0.56
blogData_50	0.03
blogData_1000	0.39
airfoilsselfnoise	0.48
mean	0.26
median	0.28

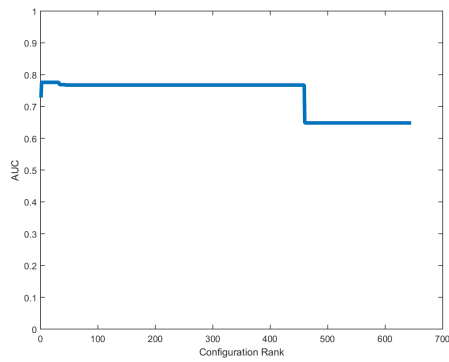
Table 4.5: Pearson's r_s between the system's resulted workflow instance ranking, and the actual one, for the regression task datasets.



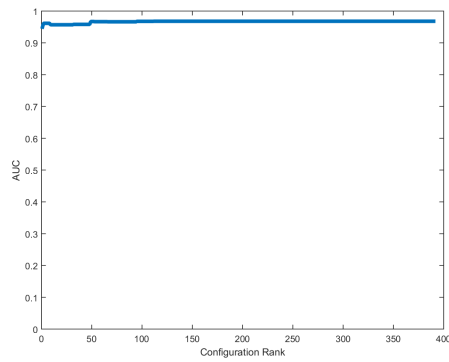
(a) ada



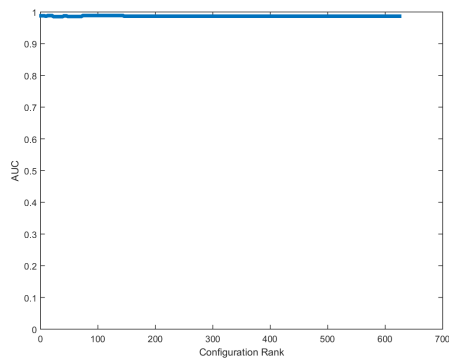
(b) gina



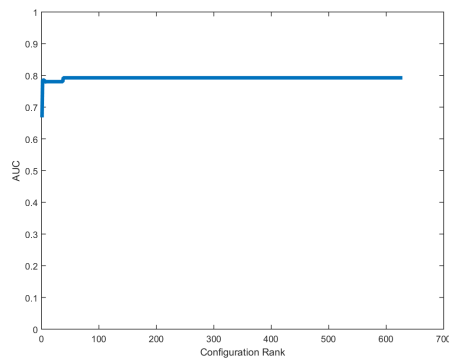
(c) hiva



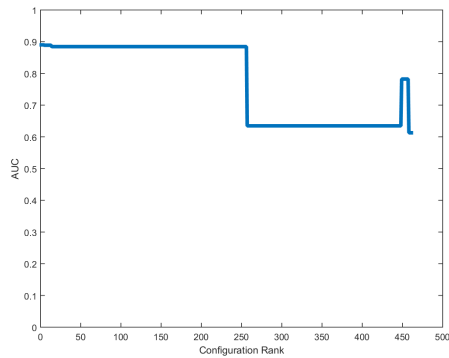
(d) farmads



(e) Nomao_1000*



(f) secom*



(g) p53_1000*

Figure 4.3: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Medium sample size Classification, (* = dataset with missing values (NaN))

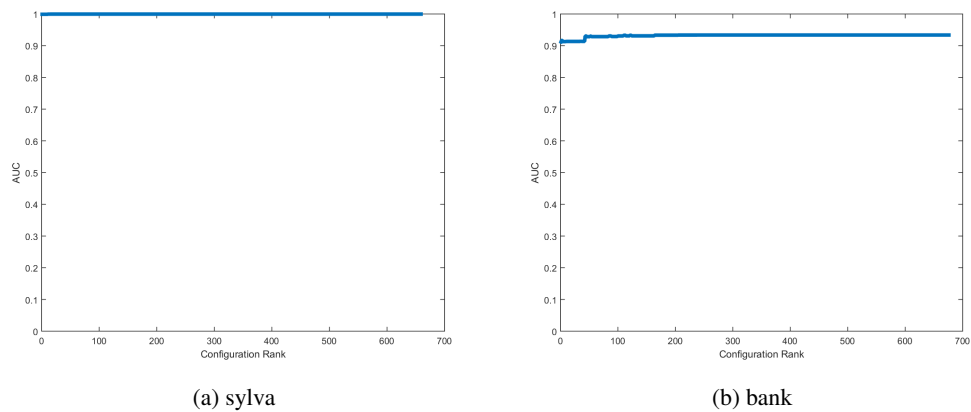
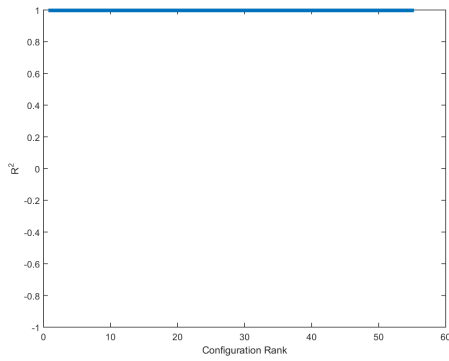
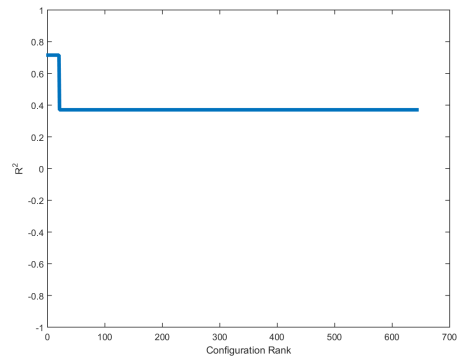


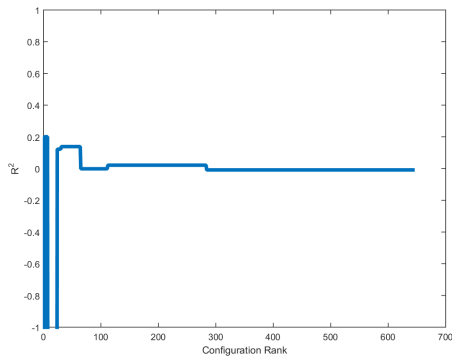
Figure 4.4: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the AUC performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Large sample size Classification



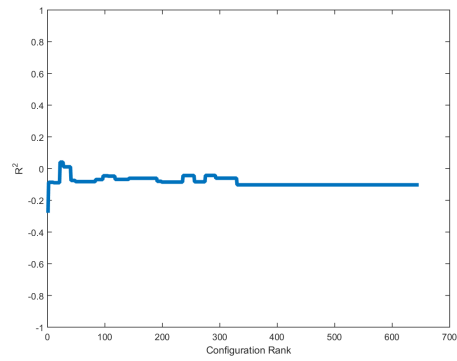
(a) yacht



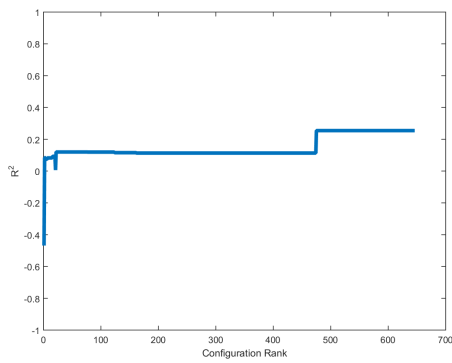
(b) cadata_50



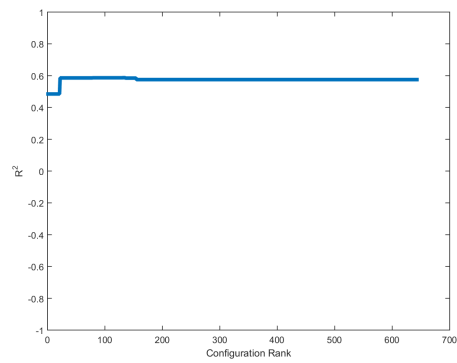
(c) blogData_50



(d) blogData_100



(e) studentmat



(f) cadata_500

Figure 4.5: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Vsmall,small sample sizes Regression

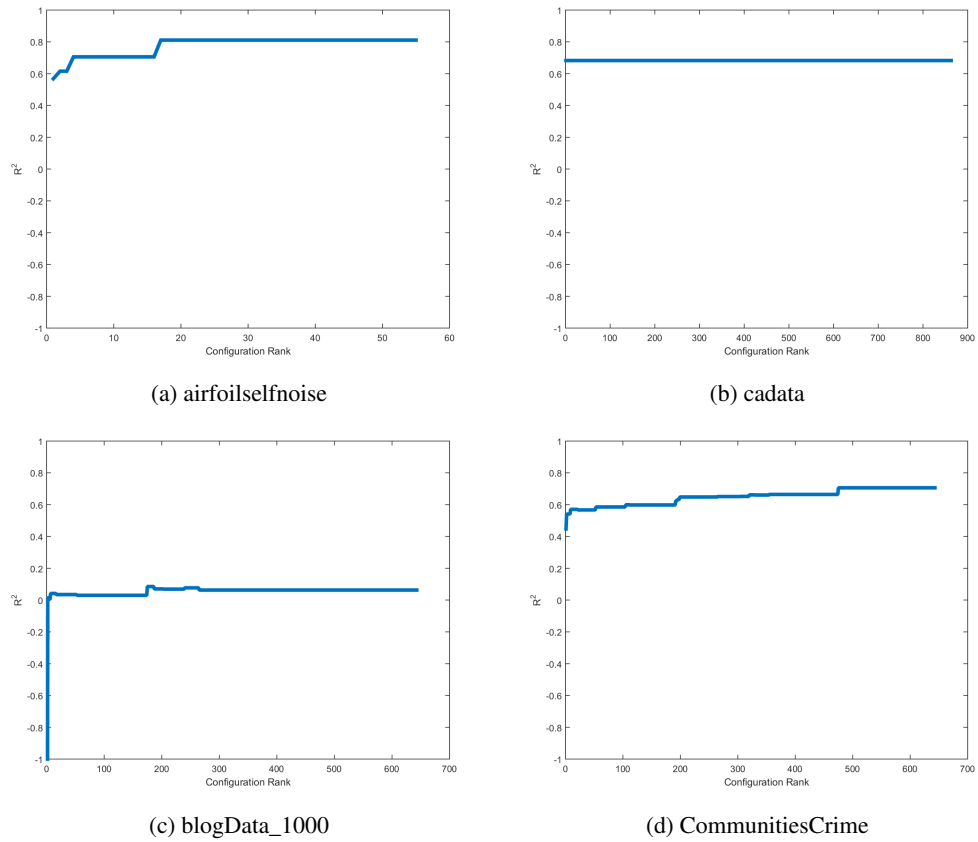
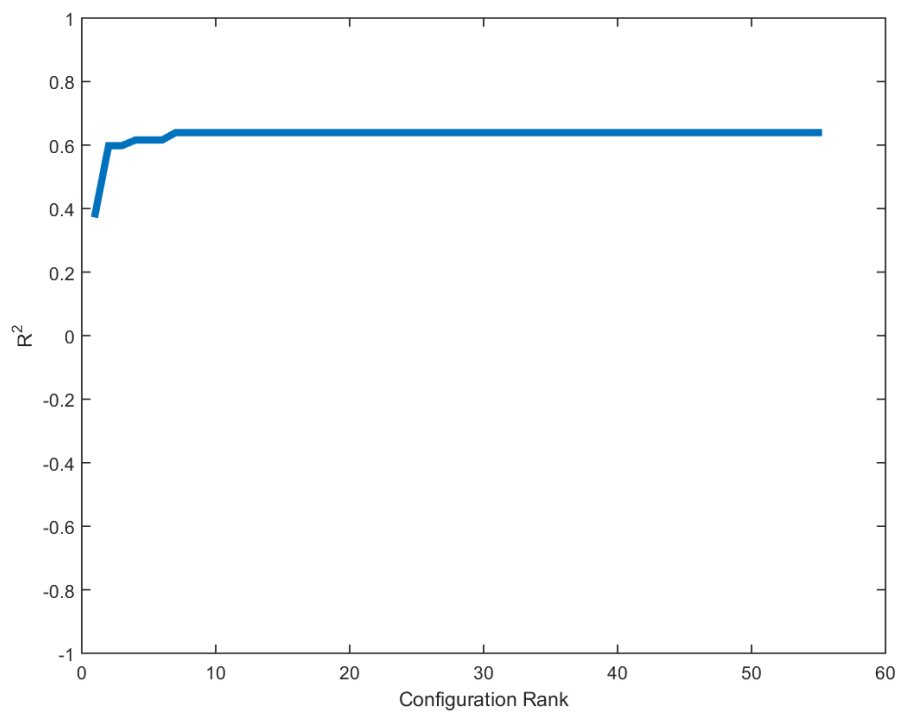


Figure 4.6: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Medium sample size Regression



(a) CASP

Figure 4.7: plot of the performance obtained from the best model, after cross validating N workflow instances, evaluated on a test set. x axes represent the R^2 performance, y axes represent the number of workflow instances executed, to obtain the respective performance. Large sample size Regression

Chapter 5

Conclusions and Future Work

In this Thesis we designed and developed an automated intelligent data mining assistant, which, based on two cooperative components, a knowledge base and a rule mechanism, composes a ranked list of proposed, appropriate, data mining workflow instances for a given dataset.

We claimed that our system can be useful, for both the experienced and the novice users, acting advisory, as a knowledge repository of up to 200 different analysis scenarios. We also advocated that our implementation enables the user to save time, by safely executing only the first small portion of the proposed series of workflow instances, while our system guarantees that the results will be the closes to best.

In order to evaluate our system and defend our claims we setup an evaluation process. We employed 31 manifold classification and regression datasets of different dimensionality and data type - we did that in order to prove that our system is capable of analyzing datasets of any kind and source. Afterwards, we used our system to generate workflow instances for these 31 datasets. Finally we executed the workflow instances on an automated system.

Two methods were employed, in order to evaluate the generated workflow instances. A Spearman's r_s , between the rank generated by our system and the true rank of the workflow instances, after executing them, and a plot for each dataset indicating the value of K (size of the portion) on which performance close to best is obtained.

The results of the evaluation, in general, proved our claims valid. In more details, the values of the classification ranking correlations (r_s) had 0.46 and 0.48 as mean and median values, which are considered as good correlations compared to the random ranking. We also concluded that the minor difference between the two values empowers the certainty that our system is equally capable of generating accurate workflow instances for any classification dataset. As for the regression ranking correlations, the results were not the best. What possibly is the main reason for this fact is that regression is a more complex and unpredictable task, compared to the simple classification task. In order to deal with this problem the system needs to incorporate more specific rules regarding the regression task and mostly related to other more descriptive characteristics of each different dataset, such as, mean correlation coefficient for any two continuous attributes, mutual information of

each discrete attribute and target class, median entropy of features, and χ^2 of discrete attributes and target class, which are of the most descriptive meta-features in the meta-learning, according to [36].

Furthermore, the K values resulting from the plots proved that indeed a user can trust our system and reliably execute only a few of the highest ranked workflow instances in order to save time, at least for the classification datasets, even though there were some datasets, such as the student, and the CommunitiesCrime which scored bad results. The student dataset is coming from the educational field, trying to predict the grade of a student on a subject. As it is claimed in [30] the decision trees are of the most efficient methods for the analysis of educational datasets. In our analysis a workflow instance containing a decision tree as the modeling method, was the one that scored the best performance. The malfunction in this case is that our system is taking into consideration general heuristics regarding any classification or regression dataset. Based on them, the decision tree workflow instances are lower scored, and thus the generation of an appropriate workflow instance ranking fails (as it is seen in Figure 4.5). In order to eliminate this, we should also incorporate domain specific rules, to enable the system, to express a rule, regarding the educational domain, which will higher score the decision trees. Gratefully, the architecture and the technologies used in the system are offered to easily express any kind of knowledge and heuristic available, so it would be a matter of time for someone to collect all the information needed, and express them as rules.

Finally, as one more future direction, more data mining algorithms and tasks, such as multi-label and multi-class classification, will be an added-value, once they are incorporated into the system.

Bibliography

- [1] P. Panov, S. Džeroski, and L. N. Soldatova, “Representing entities in the ontodm data mining ontology,” in *Inductive Databases and Constraint-Based Data Mining*. Springer, 2010, pp. 27–58.
- [2] M. Proctor, “Drools: a rule engine for complex event processing,” in *International Symposium on Applications of Graph Transformations with Industrial Relevance*. Springer, 2011, pp. 2–2.
- [3] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, “Crisp-dm 1.0 step-by-step data mining guide,” 2000.
- [4] A. Bernstein, S. Hill, and F. Provost, “Intelligent assistance for the data mining process: An ontology-based approach,” *Information Systems Working Papers Series, Vol.*, 2002.
- [5] C. Diamantini, D. Potena, and E. Storti, “Supporting users in kdd processes design: a semantic similarity matching approach,” in *Planning to Learn Workshop (PlanLearn’10) at ECAI*, 2010, pp. 27–34.
- [6] S. Fernández, R. Suárez, T. De La Rosa, J. Ortiz, F. Fernández, D. Borrajo, and D. Manzano, “Improving the execution of kdd workflows generated by ai planners,” in *Planning to Learn Workshop (PlanLearn’10) at ECAI*, 2010, pp. 19–25.
- [7] R. Engels, G. Lindner, and R. Studer, “A guided tour through the data mining jungle.” in *KDD*. Citeseer, 1997, pp. 163–166.
- [8] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, “Data mining workflow templates for intelligent discovery assistance and auto-experimentation,” *Third-Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-10)*, pp. 1–12, 2010.
- [9] S. Moro, P. Cortez, and P. Rita, “A data-driven approach to predict the success of bank telemarketing,” *Decision Support Systems*, vol. 62, pp. 22–31, 2014.
- [10] F. Serban, J.-U. Kietz, and A. Bernstein, “An overview of intelligent data assistants for data analysis,” in *Planning to Learn Workshop (PlanLearn’10) at ECAI*, 2010, pp. 7–14.

- [11] C. Diamantini, D. Potena, and E. Storti, “Kddonto: An ontology for discovery and composition of kdd algorithms,” *Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD’09)*, pp. 13–24, 2009.
- [12] M. Hilario, P. Nguyen, H. Do, A. Woznica, and A. Kalousis, “Ontology-based meta-mining of knowledge discovery workflows,” in *Meta-Learning in Computational Intelligence*. Springer, 2011, pp. 273–315.
- [13] P. Nguyen, M. Hilario, and A. Kalousis, “Using meta-mining to support data mining workflow planning and optimization,” *Journal of Artificial Intelligence Research*, vol. 51, pp. 605–644, 2014.
- [14] A. Bernstein, F. Provost, and S. Hill, “Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification,” *IEEE Transactions on knowledge and data engineering*, vol. 17, no. 4, pp. 503–518, 2005.
- [15] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, “Pddl-the planning domain definition language,” 1998.
- [16] R. Grossman, S. Bailey, A. Ramu, B. Malhi, P. Hallstrom, I. Pulleyn, and X. Qin, “The management and mining of multiple predictive models using the predictive modeling markup language,” *Information and Software Technology*, vol. 41, no. 9, pp. 589–595, 1999.
- [17] J. S. Long and J. Freese, *Regression models for categorical dependent variables using Stata*. Stata press, 2006.
- [18] S. F. Crone, S. Lessmann, and R. Stahlbock, “The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing,” *European Journal of Operational Research*, vol. 173, no. 3, pp. 781–800, 2006.
- [19] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
- [20] I. Tsamardinos, V. Lagani, and D. Pappas, “Discovering multiple, equivalent biomarker signatures,” in *7th Conference of the Hellenic Society for Computational Biology and Bioinformatics (HSCBB12)*. Heraklion, 2012.
- [21] R. Kohavi *et al.*, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, vol. 14, no. 2. Stanford, CA, 1995, pp. 1137–1145.
- [22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.
- [23] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

- [24] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.
- [25] J. Friedman and T. Hastie, “Tibshirani r. glmnet: lasso and elastic net regularized generalized linear models; 2008,” *R language*, 2013.
- [26] A. E. Hoerl and R. W. Kennard, “Ridge regression: applications to nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.
- [27] P. Waldmann, G. Mészáros, B. Gredler, C. Fuerst, and J. Sölkner, “Evaluation of the lasso and the elastic net in genome-wide association studies,” *Frontiers in genetics*, vol. 4, p. 270, 2013.
- [28] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [29] C. Strobl, J. Malley, and G. Tutz, “An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests.” *Psychological methods*, vol. 14, no. 4, p. 323, 2009.
- [30] P. Strecht, L. Cruz, C. Soares, J. Mendes-Moreira, and R. Abreu, “A comparative study of classification and regression algorithms for modelling students’ academic performance.” *International Educational Data Mining Society*, 2015.
- [31] S. F. Santibanez, M. Kloft, and T. Lakes, “Performance analysis of machine learning algorithms for regression of spatial variables. a case study in the real estate industry.”
- [32] A. Okujeni, S. Van der Linden, B. Jakimow, A. Rabe, J. Verrelst, and P. Hostert, “A comparison of advanced regression algorithms for quantifying urban land cover,” *Remote Sensing*, vol. 6, no. 7, pp. 6324–6346, 2014.
- [33] S. Bernard, L. Heutte, and S. Adam, “Influence of hyperparameters on random forest accuracy,” in *International Workshop on Multiple Classifier Systems*. Springer, 2009, pp. 171–180.
- [34] S. Kotsiantis, “Supervised machine learning: A review of classification techniques,” *Informatica*, vol. 31, pp. 249–268, 2007.
- [35] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, “A comprehensive evaluation of multiclassification methods for microarray gene expression cancer diagnosis,” *Bioinformatics*, vol. 21, no. 5, pp. 631–643, 2004.
- [36] J. W. Lee, *Relationships among learning algorithms and tasks*. Brigham Young University, 2011.