Learning the Impact of Data Pre-processing in Data Analysis

Besim Bilalli

UNIVERSITAT POLITÈCNICA DE CATALUNYA
Department of Service and Information System Engineering
Barcelona, 2018

Learning the Impact of Data Pre-processing in Data Analysis

Besim Bilalli

supervised by

Dr. Alberto Abelló

Dr. Tomàs Aluja-Banet

Dr. Robert Wrembel

Thesis submitted for the degree of Doctor of Philosophy at Universitat Politècnica de Catalunya and Poznan University of Technology, in partial fulfillment of the requirements within the scope of the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College".

May, 2018

Learning the Impact of Data Pre-processing in Data Analysis. May 2018.

Besim Bilalli
bbilalli@essi.upc.edu

Database Technologies and Information Management Group
Universitat Politècnica de Catalunya
Jordi Girona, 1-3
08034 - Barcelona, Spain

UPC Main Ph.D. Supervisors:	Dr. Alberto Abelló
	Dr. Tomàs Aluja-Banet
	Universitat Politècnica de Catalunya, BarcelonaTech, Spain
PUT Ph.D. Supervisor:	Dr. Robert Wrembel
	Poznan University of Technology, Poland
Ph.D. Committee:	Dr. Jérôme Darmont
	Université de Lyon, France
	Dr. Wolfgang Lehner
	Technische Universität Dresden, Germany
	Dr. Sergio Ilarri
	University of Zaragoza, Spain
Ph.D. Series:	Barcelona School of Informatics, Universitat Politècnica de Catalunya, BarcelonaTech

This dissertation is available on-line at the Theses and Dissertations On-line (TDX) repository, which is coordinated by the Consortium of Academic Libraries of Catalonia (CBUC) and the Supercomputing Centre of Catalonia Consortium (CESCA), by the Catalan Ministry of Universities, Research and the Information Society.

Acknowledgments

Imagine a world where everyone works alone and for himself. Such a world is apparently ought to fail. We humans are social beings that are meant to collaborate, and we progress only by working together. Therefore no one but all of us, can claim the collective human knowledge. Similarly this work, which I hope is a tiny contribution to the human knowledge was realized through collaboration. Therefore, during the last four years I had the chance to meet and work with many wonderful people. Overall, I can only say that I am grateful to have known all of them, and I would like to use this occasion to explicitly thank at least some of them.

First of all, I would like to thank Dr. Alberto Abelló and Dr. Tomàs Aluja for being such great people. Their endless support, discipline, professionalism, and patience made this Ph.D. possible. They obviously taught me many things in the professional sense, but I think I also learned, in an implicit way (I really hope I did), plenty of things in the social sense.

I would like to thank Dr. Robert Wrembel for being so kind and always helpful.

I would like to thank my cousin and friend Mennan Selimi, and my friends back home, for being always there to share the good times, and for helping me to get through the hard times.

I would like to thank Rana Faisal Munir for being such a positive and supportive person. His daily support made the daily pains feel less painful.

I would like to thank Gaston Bakkalian for making me enjoy my stay in Poznan.

I would like to thank the whole DTIM group (past and present members) for the great memories and the joyful weekly lunch seminars we shared together.

I would like to thank my whole family for the way they have always valued knowledge and learning.

I would like to thank my beloved wife Besjana, who has brought nothing but happiness to my life.

Finally, let me close with an Arabic word that expresses in a perfect way all the possible thanks to the Almighty Creator, Alhamdulilah.

Besim Bilalli
May 14th, 2018
Barcelona, Spain

* * *

This work has been funded by the European Commission (EACEA) through the Erasmus Mundus doctoral fellowship, via the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College (IT4BI-DC)".

Abstract

There is a clear correlation between data availability and data analytics, and hence with the increase of data availability — unavoidable according to Moore's law, the need for data analytics increases too. This certainly engages many more people, not necessarily experts, to perform analytics tasks. However, the different, challenging, and time consuming steps of the data analytics process, overwhelm non-experts and they require support (e.g., through automation or recommendations).

A very important and time consuming step that marks itself out of the rest, is the *data pre-processing step*. Data pre-processing is challenging but at the same time has a heavy impact on the overall analysis. In this regard, previous works have focused on providing user assistance in data pre-processing but without being concerned on its impact on the analysis. Hence, the goal has generally been to *enable* analysis through data pre-processing and not to *improve* it. In contrast, this thesis aims at developing methods that provide assistance in data pre-processing with the only goal of improving (e.g., increasing the predictive accuracy of a classifier) the result of the overall analysis.

To this end, we propose a method and define an architecture that leverages ideas from *meta-learning* to learn the relationship between transformations (i.e., pre-processing operators) and mining algorithms (i.e., classification algorithms). This eventually enables ranking and recommending transformations according to their potential impact on the analysis.

To reach this goal, we first study the currently available methods and systems that provide user assistance, either for the individual steps of data analytics or for the whole process altogether. Next, we classify the *metadata* these different systems use and then specifically focus on the metadata used in meta-learning. We apply a method to study the predictive power of these metadata and we *extract* and *select* the metadata that are most relevant.

Finally, we focus on the user assistance in the pre-processing step. We devise an architecture and build a tool, PRESISTANT, that given a classification algorithm is able to recommend pre-processing operators that once applied, positively impact the final results (e.g., increase the predictive accuracy). Our results show that providing assistance in data pre-processing with the goal of improving the result of the analysis is feasible and also very useful for non-experts. Furthermore, this thesis is a step towards demystifying the non-trivial task of pre-processing that is an exclusive asset in the hands of experts.

Keywords

data pre-processing; supervised learning; data mining; meta-learning;

Resumen

Existe una clara correlación entre disponibilidad y análisis de datos, por tanto con el incremento de disponibilidad de datos — inevitable según la ley de Moore, la necesidad de analizar datos se incrementa también. Esto definitivamente involucra mucha más gente, no necesariamente experta, en la realización de tareas analíticas. Sin embargo los distintos, desafiantes y temporalmente costosos pasos del proceso de análisis de datos abruman a los no expertos, que requieren ayuda (por ejemplo, automatización o recomendaciones).

Uno de los pasos más importantes y que más tiempo conlleva es el *pre-procesado de datos*. Pre-procesar datos es desafiante, y a la vez tiene un gran impacto en el análisis. A este respecto, trabajos previos se han centrado en proveer asistencia al usuario en el pre-procesado de datos pero sin tener en cuenta el impacto en el resultado del análisis. Por lo tanto, el objetivo ha sido generalmente el de *permitir* análizar los datos mediante el pre-procesado y no el de *mejorar* el resultado. Por el contrario, esta tesis tiene como objetivo desarrollar métodos que provean asistencia en el pre-procesado de datos con el único objetivo de mejorar (por ejemplo, incrementar la precisión predictiva de un clasificador) el resultado del análisis.

Con este objetivo, proponemos un método y definimos una arquitectura que emplea ideas de *meta-aprendizaje* para encontrar la relación entre transformaciones (operadores de pre-procesado) i algoritmos de mineria de datos (algoritmos de classificación). Esto, eventualmente, permite ordenar y recomendar transformaciones de acuerdo con el impacto potencial en el análisis.

Para alcanzar este objetivo, primero estudiamos los métodos disponibles actualmente y los sistemas que proveen asistencia al usuario, tanto para los pasos individuales en análisis de datos como para el proceso completo. Posteriormente, clasificamos los *metadatos* que los diferentes sistemas usan y

ponemos el foco específicamente en aquellos que usan metadatos para meta-aprendizaje. Aplicamos un método para estudiar el poder predictivo de los metadatos y *extraemos* y *seleccionamos* los metadatos más relevantes.

Finalmente, nos centramos en la asistencia al usuario en el paso de pre-procesado de datos. Concebímos una arquitectura y construímos una herramienta, PRESISTANT, que dado un algoritmo de classificación es capaz de recomendar operadores de pre-procesado que una vez aplicados impactan positivamente el resultado final (por ejemplo, incrementan la precisión predictiva). Nuestros resultados muestran que proveer asistencia al usuario en el pre-procesado de datos con el objetivo de mejorar el resultado del análisis es factible y muy útil para no-expertos. Además, esta tesis es un paso en la dirección de desmitificar que la tarea no trivial de pre-procesar datos esta solo al alcance de expertos.

Palabras Clave

pre-procesado de datos; aprendizaje supervisado; minería de datos; meta-aprendizaje;

Streszczenie

Jednym z głównych zagadnień związanych z analizą danych jest zapewnienie poprawnych wyników analiz i wiarygodnych modeli predykcji. Na jakość obu ma wielki wpływ jakość samych danych. Z tego względu, wyzwaniem badawczym jest odpowiednie wstępne przygotowanie danych do analizy.

W całym procesie analizy danych wyróżnia się 4 następujące zadania: selekcję danych (ang. data selection), przygotowanie danych (ang. data preprocessing), analizę / eksplorację danych (ang. data analysis / mining) i interpretację / ocenę wyników (ang. interpretation / evaluation).

Selekcja polega na wyborze danych, które będą podlegały analizie. *Przygotowanie* obejmuje: (1) czyszczenie danych (m.in., eliminowanie błędów literowych, uzupełnianie wartości brakujących, ujednolicanie wartości), (2) eliminowanie duplikatów, (3) transformowanie danych do jednolitej struktury. W procesie *analizy / eksploracji* stosuje się modele statystyczne i uczenie maszynowe do wydobywania wiedzy z danych, a ich wyniki ocenia się w korku *interpretacji*.

Z omówionych wyżej zadań w procesie analizy danych, jednym z najtrudniejszych jest przygotowanie danych. Szacuje się, że zajmuje ono 50-80% łącznego czasu przeznaczonego na projekt przepływu zadań analizy danych. Ten problem jest przedmiotem badań wielu wiodących ośrodków naukowych na świecie, jednak mimo wielu propozycji, zadanie to nadal bardzo często wymaga asysty użytkownika. Ponadto, istniejące na rynku komercyjne i otwarte narzędzia informatyczne nie wspierają użytkownika w procesie przygotowania danych w sposób właściwy dla zadanego problemu analitycznego.

W ramach niniejszej rozprawy koncentrujemy się na *uczeniu nadzorowanym* (ang. supervised learning) i wybranym jego mechaniźmie - klasyfikacji, jako technice analizy danych, dla której przeprowadzamy proces przygotowania danych. W problemie klasyfikacji, właściwe przygotowanie danych do analizy ma wpływ na jakość (trafność) modelu klasyfikacji.

Ponieważ nie istnieją żadne rozwiązania umożliwiające wybór właściwego sposobu przygotowania danych, które uwzględniałyby charakterystyki statystyczne danych i docelowy model klasyfikacji, **celem niniejszej rozprawy** jest opracowanie mechanizmów wspierających użytkownika (analityka biznesowego) w procesie przygotowania danych, w taki sposób, aby zastosowane metody czyszczenia i transformowania danych zwiększyły trafność modelu predykcji (klasyfikacji). Wynikiem działania tych mechanizmów jest ranking operacji transformacji danych, rekomendowany użytkownikowi wraz z predykcją jej wpływu na jakość modelu klasyfikacji.

Zaproponowane w rozprawie mechanizmy bazują na meta-uczeniu (ang. meta-learning). Meta-uczenie polega na stosowaniu algorytmów uczenia maszynowego na metadanych opisujących eksperymenty obliczeniowe i budowaniu ogólnego modelu reprezentującego zależności między danymi eksperymentalnymi a wynikami eksperymentów. W naszym podejściu, model ten opisuje zależności pomiędzy charakterystykami statystycznymi analizowanych danych, algorytmami klasyfikacji i jakością budowanego modelu klasyfikacji. Zgodnie z naszą najlepszą wiedzą, meta-uczenie wcześniej nie było proponowane jako mechanizm wspierający proces przygotowania danych do analizy.

Cel rozprawy został zrealizowany w postaci następujących zadań, stanowiących **kontrybucję naukową rozprawy**.

- Opracowano taksonomię metadanych wykorzystywanych w procesie odkrywania wiedzy z danych i zidentyfikowano niezbędny zbiór metadanych, umożliwiający rekomendowanie właściwych technik wstępnego przygotowania danych do analizy. Zaprojektowano i zaimplementowano architekturę repozytorium metadanych.

- Opracowano nowe metody wsparcia analityka biznesowego w procesie przygotowania danych do analizy. W tym celu, zastosowano koncepcję meta-uczenia do wyboru takich metod przygotowania danych, których zastosowanie zwiększy trafność wybranych algorytmów klasyfikacji, dla zadanych charakterystyk danych wejściowych.

- Zaproponowano i zaimplementowano technikę redukcji charakterystyk wejściowego zbioru danych, tylko do tych, które mają największy wpływ na jakość modelu klasyfikacji.

- Zaprojektowano i zaimplementowano architekturę prototypowego systemu wspierającego analityka biznesowego w procesie przygotowania

danych. System ten, na podstawie charakterystyk danych wejściowych i zadanego algorytmu klasyfikacji, rekomenduje metody czyszczenia i transformowania danych, uporządkowane zgodnie z ich wpływem (pozytywny, neutralny, negatywny) na jakość modelu klasyfikacji.

- Dokonano szczegółowej oceny eksperymentalnej prototypowego systemu dla 5-ciu popularnych algorytmów klasyfikacji (tj. J48, Naive Bayes, PART, Logistic, IBk) i w oparciu o ponad 500 zbiorów danych testowych repozytorium OpenML.

Contents

1

Introduction

1.1 Background and Motivation

Today, data is treated as a powerful asset. Therefore, there is an overall boost in collecting it from everywhere and in any form. Data collection has reached such extremes that according to some, currently, in two days we generate more data than what we have generated from the dawn of civilization up until 2003[1].

In fact, data has been a valuable artifact for a long time (mainly for visualization, e.g., see Minards Work 19th century [31]), yet its value has reached the peak today, because only today we are able to generate more (actionable) knowledge from it. Companies, using data, are able to predict the next hurricane [25], the next maneuver of a human driver [62], the health diagnosis of a patient [64], flight delays[2], and many other things.

However, in general our capability of analyzing data, lags far behind the capability of collecting it. This is due to the fact that data analytics consists of several challenging and time consuming steps, which have been grouped into the following [27]: *data selection*, *data pre-processing*, *data mining*, and *in-*

[1] https://techcrunch.com/2010/08/04/schmidt-data

[2] https://www.nytimes.com/2018/02/13/travel/new-google-tips-and-tools-for-travelers.html

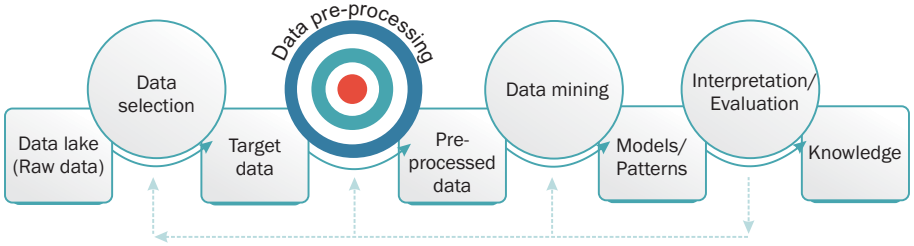


Fig. 1.1: *Data analytics* process, sometimes synonymously referred to as *knowledge discovery* or *knowledge discovery in databases (KDD)*; adapted from [27]

terpretation/evaluation (cf. Figure 1.1).

Briefly, *data selection* represents the task of sifting out the data that may not be relevant for the analysis. *Data pre-processing* represents the broad task of cleaning/wrangling the data, such that it is ready for the analysis (e.g., mining). Next, *data mining* is the task of applying a machine learning/statistical modeling algorithm on top of the pre-processed data (e.g., supervised learning, unsupervised learning). Finally, *interpretation* is the task of interpreting the results.

If we use the analogy of baking a cake, selection translates to the process of picking the ingredients (from what is already available in the kitchen), pre-processing includes the tasks of preparing the dough, the cream, and the dressing. Mining is the process of baking, and finally, interpretation is the process of tasting.

As one can imagine, one of the most time consuming steps and also the one that has a heavy impact on the final result is the pre-processing step. Yet, in contrast to baking, in data analytics, there are no clear and strictly defined "recipes" for pre-processing the data, such that the final result is improved. Hence, this step is generally performed by experts (i.e., chefs in our analogy).

But, given the availability of data (e.g., repositories like OpenML [92], UCI[3], and web APIs like, Twitter API[4], Facebook API[5]), even non-expert users want to participate in analytics tasks. However, the staggeringly large amount of pre-processing options and mining (i.e., algorithms) alternatives, overwhelm non-experts and they require support.

Many previous research efforts and practical solutions have tried to tackle specifically the problem of user support in data pre-processing [44, 49, 66,

[3]https://archive.ics.uci.edu/ml/index.php
[4]https://developer.twitter.com/en/docs/api-reference-index
[5]https://developers.facebook.com/docs/apis-and-sdks

76] — which is also the focus of this thesis, and generally the problem of automating the whole data analytics pipeline [28, 70, 71, 89] — sometimes referred to as *AutoML*. For data pre-processing, from the practical solutions side, the overall goal has been to provide off the shelf tools and packages (i.e., readily implemented algorithms) that facilitate the application of different techniques on top of data, yet still requiring user expertise (in our analogy this would be the kitchen tools for chopping, mixing, baking, etc.). From the research side, the overall goal has been to develop more sophisticated methods for providing user support e.g., recommending alternative pre-processing operators (in our baking analogy this translates to, for instance, suggesting the use of an electric mixer instead of a manual one).

The latter types of user support are very much appreciated, however, they lack in providing customized user support with the aim of improving the final results. That is to say, what is missing is to recommend pre-processing operators that would improve the final results of the analysis (in our analogy, that would be applying grandma's tips to make more delicious cakes). This implies dropping the assumption that only experts should analyze data (i.e., not only chefs are baking cakes), and opening space for more effective support, useful for non-experts.

1.2 Scope

It is almost a fact that data pre-processing consumes 50-80% of analytics time [69]. This is mainly because data pre-processing encompasses a broad range of complex activities.

There is sometimes ambiguity in the naming of the concept of data pre-processing, since many different terms have been used in practice to refer to the same process. Some of the concepts that have been interchangeably used are the following: Extract-Transform-Load (ETL), data wrangling, data cooking, data cleaning, data preparation, data transformation. There is no strict line that clearly separates them (i.e., they overlap in functionality), hence the confusion in naming.

Furthermore, the specific techniques used within this process have also been referred to, in many ways, such as: data pre-processing operators, pre-processing algorithms, transformations, filters.

To put things into perspective and to contextualize our use of the term, in Figure 1.2, we depict the overlap that different pre-processing types may

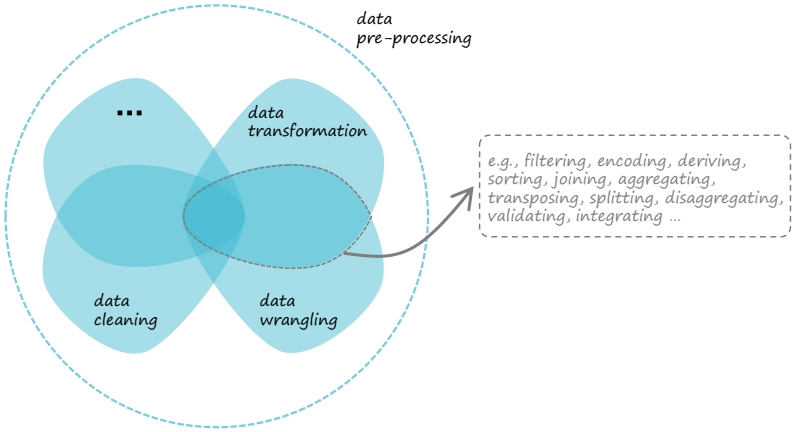


Fig. 1.2: The overlap of data pre-processing types in terms of *functionality*

have.

Such overlap is mainly within their functionality (i.e., the techniques they use), hence they may differ along some other dimensions, such as the type of users, the form of data they expect (e.g., structured, unstructured), their use case, their goals, etc. However, the boundaries of the differences are quite blurry, because they very much depend on the perspective of comparison (i.e., subjective). Furthermore, because of the overlap in terms of functionality, different pre-processing types also have overlapping behaviour in the light of data analysis. That is to say, they may all impact the results of data analysis, even if in the first hand they are not applied with such an aim. Moreover, the knowledge required to use such techniques may vary between *domain knowledge* and *expert knowledge*. The line here is not very clear too.

For a better understanding, in Table 1.1, we attempt to classify different pre-processing types along two main dimensions: i) the general characteristics in the light of pre-processing itself, and ii) characteristics in the light of the overall analysis (i.e., mining). The table does not aim to be exhaustive, but it aims to convey the idea that different data pre-processing types have no clear and sharp lines in terms of their differences, and they all, one way or another may impact the analysis.

Throughout this thesis, we consider as data pre-processing operators all the techniques that may have impact on the analysis, without distinguishing their type or category and throughout this document, we interchangeably refer to them as transformations.

Table 1.1: Comparison/classification of pre-processing types

General data pre-processing characteristics				Characteristics in the light of data analysis	
Pre-processing type	Required knowledge	Users	Goal	Obligatory for the analysis	Impact on the analysis
ETL/Data transformation	Expert	Data scientists or IT users	Transform data to the require format (reporting)	Obligatory or Optional	Possibly yes
Data wrangling	Domain	Business users	Make data more appropriate and valuable	Optional	
Data cleaning	Domain or Expert	Data scientists	Detect and clean errors	Optional	
Data integration	Expert	IT users	Combine and augment	Obligatory or optional	
Data conversion	Expert	Data scientists or IT users	Convert from one format to another	Obligatory or Optional	

Furthermore, within this thesis, the data mining (analysis) step refers to a specific method of analyzing data, which is known as *supervised learning* [81]. The task of supervised learning is: given a training set of N example input-output pairs $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$, where each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f. Here x and y are *variables* (or synonymously referred to as features or attributes) that can be of any type. Typically, x is a vector/set of variables and they are called *predictors* or *explanatory* variables, whereas y is generally a single variable that is called the *response* variable. Furthermore, when y is of continuous type the problem is referred to as *regression* and in the case when it is of categorical type the problem is referred to as *classification*. Indeed in this thesis, we focus on classification problems, since they are more widely used in practice.

1.3 Research Problems and Challenges

With the dramatic decrease of the price of data and the availability of off-the-shelf tools for data analysis, the division line between data analysts (experienced users) and everyone else (non-experienced users) is becoming thinner. Everyone is getting more and more engaged in analyzing some kind of data.

Take for instance a simple example of choosing the restaurant to go for dinner. You may go to the closest one, you may ask your friends on Twitter, or you may end up reading plenty of reviews and different kinds of information on different platforms (e.g., Yelp, TripAdvisor). Another example would be that of choosing the journal for publishing an article. There are a lot of journals and you can decide to retrieve more information (e.g., using the dblp API[6] and the Scopus[7] repository) about them in order to make your decision. These are examples of simple available data, which in addition with the endless amount of publicly available repositories (e.g., Open data initiatives[8]), bring all of us closer than ever to valuable data that one way or another can be analyzed.

On the other side, complementary to data, there is the abundant set of tools and languages (e.g. Weka, RapidMiner, R, python notebooks, AmazonML), that assist users to perform the required analysis. Hence data and tools together, engage even non-experts to perform analysis tasks. Therefore, sooner than later most of the people who analyze data will not be statisticians (i.e., experts). This revolution demands a new way of thinking and implies that more time needs to be spend on automating — providing user support, and building software.

At this point, the **first research challenge** is *to assess what are the current methods used to provide user support in the whole knowledge discovery process, and to identify and classify the data (metadata) they use to enable such support? Moreover, whether the data/metadata used are complete or is there something unexploited? Finally, how good such metadata are for providing user support in the analysis step?*

Even in the presence of many tools that make it easier for non experts to analyze their data, the data pre-processing step is still the one that consumes most of the analytics time. The currently dominating methods with respect to user support in data pre-processing either enable pre-processing by providing the (readily implemented) necessary tools for applying different techniques, or in the best case they provide recommendations that are "syntactically" valid. That is, if the data is in a form that a data pre-processing operator can be applied to it, provide it as an option/recommendation.

The problem with these approaches is that they assume the user knows which pre-processing operators to apply. Hence, they are aimed towards more experienced users, and they fail to provide support to users that do not

[6]https://dblp.uni-trier.de
[7]https://www.elsevier.com/solutions/scopus
[8]https://okfn.org/opendata

know how the data should look like (e.g., be transformed) in order to yield better analysis.

Therefore, the **second research challenge** is *to define methods that enable data pre-processing support with the aim of improving the analysis. That is, not only checking if operators are "syntactically" valid for a dataset, but also if they have some positive impact on the final analysis.*

1.4 Hypothesis

In a classification problem, pre-processing should be applied only as long as it is useful for the analysis. However, since there are no clear recipes, there is lack of user support in pre-processing with the aim of improving the classification performance (e.g., decrease the classification error). Our hypothesis is that *meta-learning* [11] can be used to provide such support, where pre-processing operators can be recommended according to their impact on the final classification performance.

This hypothesis is supported by the fact that meta-learning has already shown to be useful for different purposes [59]. For instance, in its inception it has shown to perform well in the *model-selection problem* [10, 11, 46, 79], where users are supported to select the best classifier for their problem at hand. Next, it has shown to be useful in finding *optimal workflows* for the complete data analytics process, and it has been referred to as meta-mining [40, 70]. Recently, meta-learning has also shown to provide good heuristics (i.e., to find a good seed for the optimization problem) for finding the *optimal hyper-parameters* in the *CASH problem* [28, 29, 78] — combined algorithm selection and hyperparameter optimization problem.

However, we note that meta-learning has never been used before for providing user support specifically in the data pre-processing step.

1.5 Contributions

The first research challenge, as mentioned above, consisted of first identifying and classifying the metadata that were used by different methods that aimed at providing intelligent support along the knowledge discovery process, be it as support for each step separately or the whole process altogether. Next, the challenge was about studying how good this metadata were, and how to select and extract only the most relevant metadata.

As a response to this challenge, *we assessed the currently available tools and methods with respect to the type and scope of user support they provide. We thoroughly analyzed the metadata enabling such support, and we performed a comprehensive metadata classification, identifying important metadata that was overlooked by the current tools. We developed a metadata repository to store such metadata. Furthermore, in order to study the goodness of metadata, we applied a method for analyzing their predictive power. Using this method we were able to select the most relevant metadata for a given scenario. Finally, we developed a new way of visualizing the relationships between different metadata.*

Our second research challenge was specifically related to the problem of user support in the data pre-processing step. To be more precise, the challenge was about defining new methods for providing user assistance in data pre-processing. Therefore, our response was to develop a method with such an aim. To this end, *we developed a method that leveraging ideas from meta-learning is able to provide support with the aim of improving the analysis (e.g, the mining algorithm yields better results on the pre-processed dataset) and decreasing the amount of time spent in pre-processing*. It is a method that for the first time does not aim at providing pre-processing support only for the sake of pre-processing, but instead, it is oriented towards positively contributing to the result of the analysis. We implemented a prototype, PRESISTANT, that given a classification algorithm (i.e., a meta-model is built per classification algorithm) recommends pre-processing operators that are ranked according to their impact on the final result of the analysis, enabling even non-experts to participate in the tedious task of pre-processing.

1.6 Thesis Overview

The main focus of this thesis is to provide user support in the data pre-processing step with the aim of improving the final results of classification problems. Our method to provide such support is based on meta-learning, which is the task of "learning on top of learning problems". Even though in the meta-level, this is still a learning problem, which naturally involves all the steps of the knowledge discovery process. Hence, in order to make it work we had to carefully go through all the steps of knowledge discovery. We use these steps to also guide the flow/structure of our thesis. Hence in Figure 1.3, we show the structure of the thesis in the form of a knowledge discovery process, where each phase (i.e., chapter) of our work corresponds to a step of the knowledge discovery process.

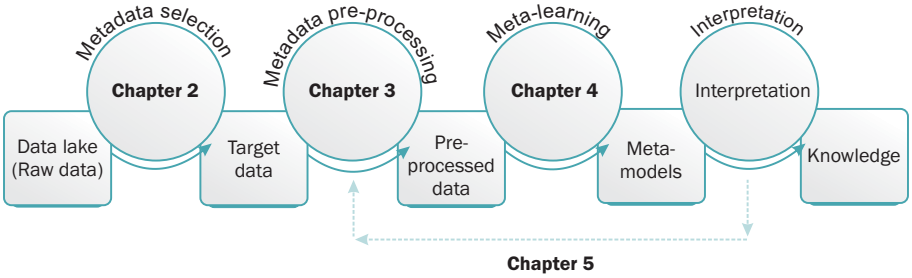


Fig. 1.3: The structure of the thesis and how it maps to a knowledge discovery process

The main chapters (2-5) of this dissertation are based on the results reported in the following publications:

P1. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, Robert Wrembel. Towards Intelligent Data Analysis: The Metadata Challenge. In: *International Conference on Internet of Things and Big Data, (IoTBD 2016)*. pp. 331-338 [Short paper]. DOI: http://dx.doi.org/10.5220/0005876203310338

P2. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, Robert Wrembel. Automated Data Pre-processing via Meta-learning. In: *International Conference on Model and Data Engineering, (MEDI 2016)*. pp. 194-208. DOI: http://dx.doi.org/10.1007/978-3-319-45547-1_16

P3. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, Robert Wrembel. Intelligent Assistance for Data Pre-processing. In: *Computer Standards &*

Interfaces, (CSI 2017). 57: 101-109. DOI: https://dx.doi.org/10.1016/j.csi.2017.05.004

P4. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet. On the predictive power of meta-features in OpenML. In: *Applied Mathematics and Computer Science, (AMCS 2017).* 27(4): 697-712. DOI: https://dx.doi.org/10.1515/amcs-2017-0048

P5. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, Robert Wrembel. Learning based recommending assistant for data pre-processing. In: *eprint arXiv: https://arxiv.org/pdf/1803.01024.pdf* [Under review].

P6. Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, Rana Faisal Munir, Robert Wrembel. PRESISTANT: Data Pre-processing Assistant. To appear in: *International Conference on Advanced Information Systems Engineering, (CAiSE 2018)* [Demo paper]. DOI: TBD

Each chapter corresponds to a publication. Chapter 2 corresponds to **P1**, Chapter 3 corresponds to **P2** and **P3**, Chapter 4 corresponds to **P4**, and Chapter 5 corresponds to **P5**. The related work is performed for each chapter separately and included inside the respective chapters. In addition, Appendix A corresponds to **P6**, where we discuss the implementation of our prototype tool. Finally, in Appendix B we show additional results for the classification algorithms that were considered but not reported in Chapter 5. In the following, we provide an overview of each chapter.

1.6.1 Chapter 2: Towards Intelligent Data Analysis: The Metadata Challenge

The advances in data storage and data collection led to the need for developing languages and frameworks that enabled performing knowledge discovery. Yet because of the intrinsic complexities and challenges faced during all the phases of the knowledge discovery process, these frameworks fell short in providing the required assistance. Therefore, there was need for developing more sophisticated methods and tools that would make knowledge discovery easier and more practical. Hence, the appearance of the concept of *intelligent data analysis*, which referred to the methods that aimed at facilitating the application of knowledge discovery.

In Chapter 2, we survey the different methods and tools that were developed to provide intelligent user support. The tools serving such a purpose

are referred to as Intelligent Discovery Assistants [6], and in order to provide the required support they make use of different meta information (metadata). In this chapter, we identify and classify all the metadata used for providing user support. We study the metadata with respect to their *roles* and *types* in an IDA. Furthermore, we identify metadata that have been overlooked and if used could provide further benefits, e.g., in the presence of metadata about the domain one could have business understanding of the available data and this may further be used by the system to assist users in easily selecting the target data (i.e., support for the first step of the analysis). Moreover, we develop a metadata repository that can store such metadata and we also provide a first attempt for an architecture that exploits such a metadata repository to further advance the user support in knowledge discovery. Therefore, the work of this chapter basically corresponds to the first step of the knowledge discovery process shown in Figure 1.3, with the only difference that instead of data selection in this case we are concerned with metadata selection. Thus, here we discuss all the possible metadata that can be used in the following steps.

1.6.2 Chapter 3: On the Predictive Power of Meta-features in OpenML

Different methods developed for providing user support in knowledge discovery (i.e., data analytics) enabled that this process does not exclusively remain in the hands of expert users, but that it can also be used by non-experienced users.

One particular method that enabled user support, specifically in the data mining step was *meta-learning* [11]. Assuming that users dealt with a classification problem — i.e., for a given a dataset they had to choose a classification algorithm and there is no best for every situation [94], meta-learning enabled support by recommending a classification algorithm that would best fit in the particular situation. In short, meta-learning enables support by collecting dataset characteristics and performance measures of different algorithms on datasets and learning a model (i.e., predictive model) on top of such historical metadata. This model is then used to provide support for newly arriving datasets.

In Chapter 3, we use a method to perform exploratory analysis on top of metadata used in meta-learning. As such, this chapter naturally maps to the pre-processing step shown in Figure 1.3, and therefore is necessary for the

next steps.

Our method consists of two phases, namely *extraction* and *selection*. In the extraction phase latent concepts are generated out of metadata, and this allows to study the metadata in a higher abstract level providing benefits such as being more generic. In the second phase, exploring the relationship between the latent concepts, the method selects the most relevant latent features. This enables the reduction of the number of latent concepts that need to be extracted and allows to have more robust models. Furthermore, in this chapter we devise a new way of visualizing the relationships between different metadata.

To put our method in practice, we first developed a tool that can retrieve metadata from OpenML — one of the biggest repositories for such metadata, and then on top of them we applied our method. We were able to study the OpenML metadata in the latent level and the experiments showed that our method was capable of selecting the metadata with more predictive power. More precisely, with the selected metadata we were able to obtain better and more robust predictive models in meta-learning, compared to the models built without applying our method.

1.6.3 Chapter 4: Intelligent Assistance for Data Pre-processing

Several factors impact the success of a given analytics task. First of all, the main important factor is the goodness (i.e., quality) of data. In a classification problem, this would translate to having a dataset that consists of relevant features/attributes and complete instances/rows. That is, features are engineered using expertise from the domain, and the instances do not contain missing values or other inconsistencies.

The second important factor determining the success of the analytics is the mining algorithm used. In a classification problem, this would be the classification algorithm and its parametrization. Thus, given the right dataset and the right mining algorithm, one can obtain good (i.e., optimal) results in an analytics task. However, in practice this is rarely the case. Typically, either the dataset is not in the proper shape or there is lack of expertise in applying the right algorithm. The problem aggravates with non-experienced users. A solution for such cases would be either to provide user support when selecting the classification algorithm, or providing assistance in transforming the dataset (data pre-processing) such that it yields better results in the analysis. For the former, in the previous chapter we argue that meta-feature extrac-

tion and selection may help on building better models for assisting users on choosing the right classification algorithms. For the latter, we propose a solution in this chapter. Hence, in this chapter, we specifically tackle the problem of user support in data pre-processing.

Previous works have generally aimed at providing assistance in data pre-processing, agnostically to its impact on the final result of the analysis. That is, the goal has been to *enable* the analysis and not to *improve* it. In this chapter, we discuss the possibility of providing user assistance in data pre-processing with the only goal of improving the final analysis. To this end, we propose a method and define an architecture that leverages ideas from meta-learning in order to learn the relationship between transformations (i.e., pre-processing operators) and classification algorithms. This, eventually enables proposing transformations according to their relevance to the analysis.

Clearly this chapter maps to the learning (mining) step of the knowledge discovery process, and because it is performed on top of metadata, it is coined under the term meta-learning (i.e., meta-mining) in Figure 1.3. Specifically, here the models are built/learned on top of meta-features (metadata) selected using the method from the previous chapter. These meta-features are extracted from transformed datasets and they are used as input for predicting the *performance* of classification algorithms on the transformed datasets. The *performance* of classification algorithms are evaluated using: *predictive accuracy*, *precision*, *recall*, and *area under the roc curve (AUC)*. Hence, the predicted values are of continuous type (i.e., numbers) and therefore our meta-learning problem translates to a *regression* problem.

In this chapter, we evaluate our approach on hundreds of datasets retrieved from OpenML and show that as an initial approach, this method provides promising results. The results obtained are statistically significant. More importantly however, the achievement of this chapter is in showing that it is possible to recommend data pre-processing operators that ultimately improve the analysis. This means that for the first time data pre-processing is not treated as an independent or isolated step within the knowledge discovery process.

1.6.4 Chapter 5: Learning Based Recommending Assistant for Data Pre-processing

The fact that data pre-processing impacts the results of the analysis is unquestionable. However, there is not much empirical study on how different

pre-processing operators impact the performance of different mining algorithms (e.g., classification algorithms). Furthermore, it is very challenging, yet very useful, to quantify the impact without explicitly applying the mining algorithms on the transformed datasets. At best, this can be done roughly by domain experts.

In this chapter, we build on top of our work explained in the previous chapter, where we argue that meta-learning can be used to push the user support in the data pre-processing step. This time in addition, we empirically study the impact of data pre-processing operators on different classification algorithms. Our study leads to a better understanding of the relationship between data pre-processing operators and classification algorithms, which in turn leads towards defining heuristic rules that can be used to reduce the search space. We redefine our architecture in order to take into account such rules, which can be further extended with expert rules, with the goal of introducing some domain knowledge.

Differently from the previous chapter, this time the *impact* of pre-processing operators is defined as the change induced on a performance measure of a mining algorithm. That is to say, in classification problems we compute the relative change that a transformation causes on the predictive accuracy of a classification algorithm on a given dataset. This can be *positive*, *negative*, or *zero*. Then, a predictive meta-model is learned to predict the *impact*, given the characteristics of a transformed dataset. The predictions ultimately enable ranking the transformations and recommending the most relevant ones to the user.

Since this time, the response feature (i.e., the feature to be predicted) is the relative change, it can be encoded as a categorical variable (i.e., with three categories: positive, negative, and zero) and thus, the meta-learning problem translates to a classification problem. Another difference from the previous chapter is that, this time, to the set of meta-features considered (i.e., dataset characteristics), we attach also the base performance of the classification algorithm (i.e., the performance before the transformation is applied). In addition, we add features that capture the difference between the meta-features before and after transformations are applied. We call these features *delta meta-features*. As a result, every meta-feature has its corresponding delta meta-feature. Indeed, the additional computational cost that is induced because of the new features attached (i.e., the extraction cost increases), compensates, because of the better results obtained by this approach in compari-

son to the previous, explained in Chapter 4.

We built a tool, PRESISTANT, to demonstrate our method and we extensively evaluate its performance both from the meta-learning perspective — how accurate predictions are, and the user perspective — what is the gain obtained from the recommendations of PRESISTANT. Within this chapter only results with respect to one algorithm (i.e., Nearest Neighbor) are shown. The results obtained for the other algorithms (i.e., Decision Tree, Naive Bayes, Logistic, and PART) are described in Appendix B. Furthermore, details on the implementation of PRESISTANT can be found in Appendix A.

We note that this chapter is a result of the interpretation and evaluation performed, which required looping back to the pre-processing (i.e., metadata pre-processing) step (cf. Figure 1.3) in order to design better features and redo meta-learning by considering these new features too.

2

Towards Intelligent Data Analysis: The Metadata Challenge

Once analyzed correctly, data can yield substantial benefits. The process of analyzing the data and transforming it into knowledge is known as Knowledge Discovery in Databases (KDD). The plethora and subtleties of algorithms in the different steps of KDD, render it challenging. An effective user support is of crucial importance, even more now, when the analysis is performed on Big Data. Metadata is the necessary component to drive the user support. In this chapter, we study the metadata required to provide user support on every stage of the KDD process. We show that intelligent systems addressing the problem of user assistance in KDD are incomplete in this regard. They do not use the whole potential of metadata to enable assistance during the whole process. We present a comprehensive classification of all the metadata required to provide user support. Furthermore, we present our implementation of a metadata repository for storing and managing this metadata and explain its benefits in a real Big Data analytics project.

2.1 Introduction

Our capability of gathering data has developed to the highest extents, whereas the ability to analyze it, lags far behind. Storing huge volumes of data is worth the effort only if we are able to transform data into knowledge. The process of transforming data into knowledge is known as Knowledge Discovery in Databases (KDD) — synonymously referred to as knowledge discovery or data analytics, cf. Figure 1.1.

The need for knowledge discovery is rising mainly thanks to the low-cost, distributed data storage and processing platforms (e.g., Apache Hadoop[1]). They allow storing and processing huge datasets on large clusters of commodity hardware. A Data Lake, for instance, is an important component of the data analytics pipeline in the world of Big Data. The idea is to have a single store of all the raw data (e.g., structured and unstructured) that anyone in an organization might need to analyze. However, the relevant data over which the analysis is going to be performed needs to be selected from the whole range of the available data. As the selection of data affects the results of the analysis, data needs to be thoroughly tracked in order to justify the results (e.g., lineage). The representation and the quality of data also affect the analysis. Raw data is often irrelevant, redundant, and incomplete and requires pre-processing. Once the data is pre-processed, there comes the difficult task of selecting the most adequate mining algorithm for a given problem. Many different algorithms are available and their performance can vary considerably. After data mining, the evaluation/interpretation step follows. The generated models need to be interpreted and/or evaluated to be understood by the user.

All in all, the above mentioned steps indicate that knowledge discovery in general is an inherently challenging task. Therefore, users need to be thoroughly supported. A lot of research has been done in this regard and systems that aim at providing user assistance have been developed. These systems are referred to as Intelligent Discovery Assistants (IDAs) [6]. The driving factor for the user assistance is the metadata they consider. Yet, there is no agreement on which kinds of metadata need to be gathered and stored in order to provide user assistance. In this chapter we tackle the problem by studying the types and roles of metadata. We observe that the meta knowledge considered in IDAs is not complete (e.g., domain knowledge and lineage

[1] https://hadoop.apache.org

is missing). Hence, we provide a classification of the metadata needed to support the whole process and discuss the implementation of our metadata repository.
Contributions. In particular, our main contributions are as follows.

- We identify and extend the metadata required for providing user support for the whole process of KDD including the very first step of data selection and we provide a classification of these metadata.
- We implement a metadata repository with the aim of storing and managing the metadata discovered and show its benefits in a real case scenario.

The rest of the chapter is organized as follows. Section 2.2 presents an analysis of IDAs and briefly discusses the differences between different categories of these systems. Section 2.3 studies the metadata required for providing user support and shows examples of systems using the respective metadata. Section 2.4 contributes a classification of the metadata needed to support the whole process of KDD. Section 2.5 shortly presents the implementation of our metadata repository and its benefits in a real Big Data analytics project. Section 2.6 discusses the related work. Finally, Section 2.7 concludes the chapter.

2.2 Intelligent Discovery Assistants

The KDD process is challenging for novice users. As already stated in Section 2.1, the most prominent works done in terms of providing helpful assistance to the users are through IDAs. In order to complete our study on the metadata needed for the user support we have to know how and to what extent these metadata are used by different IDAs. Depending on the core techniques and metadata used, IDAs can be divided into 5 broad categories [82], namely: *expert systems*, *meta-learning systems*, *case-based reasoning systems*, *planning-based data analysis systems*, *workflow composition environments*.
Expert systems (ES) are the earliest and the simplest systems to provide help to the user during the data mining phase. Their main component is a knowledge base consisting of expert rules, which determine the mining algorithm to be used. Questions are posed to the user about a given problem and the information provided as response is used by the system in order to assess which rule is appropriate.

Meta-learning systems (MLS) are more advanced. The rules that were statically defined by the experts in the previous category are dynamically learned here. MLSs try to discover the relationship between measurable features of the dataset and the performance of different algorithms, which is a standard learning problem. The learned model can then be used to predict the most suitable algorithm for a given dataset.
The idea behind *case-based reasoning systems* (CBR) is to store the successfully applied workflows as cases, in a *case base*, with the only goal of reusing them in the future. When faced with a new problem (i.e., dataset) provided by the user, these systems return k previous cases from the case base according to the level of similarity with the new problem. The selected workflow can then be adapted to properly fit and solve the new problem. Their disadvantage, as in MLSs, is that they can provide structured help only if a new problem is similar to the problems seen so far.
Planning-based data analysis systems (PDAS) are able to autonomously design valid workflows without relying on the similarity between different problems. In order to do this, the workflow composition problem is seen as a planning problem, where a plan is built by combining operators that transform the initial problem into accurate models or predictions. In order to construct valid workflows, the input, output, preconditions, and effects of each operator need to be known. Once the conditions are met, operators are composed to form valid but not necessarily optimal workflows, which at a later stage are ranked.
Workflow composition environments (WCE) do not provide automatic support for data analysis, but facilitate the use of different data mining algorithms providing nice graphical environments for quick workflow design and execution.

2.3 Metadata Challenge in KDD

In this section, we analyze what can be achieved by collecting metadata and what kinds of metadata can be collected in a KDD environment.

2.3.1 The Role of Metadata

The generation and management of metadata can determine the type of support offered. We differentiate among the following.
Single-step support. It is an indication of the complexity of the advice of-

fered. The single step for which some kind of user support or even automation is provided is usually the data mining step of the KDD process.
Multi-step support. Similarly, it indicates the complexity of the advice offered. Metadata can be used to extend the support to several steps of KDD.
Variable selection support. It indicates whether a system provides user support in the very first phase of a KDD process. It is of crucial importance when an analysis of raw data needs to be done (e.g., in a Big Data environment). Raw data in this context refers to data that is not offered in a form of a datase, but it is stored in its original format. Hence, prior to analysis, the data of interest needs to be selected and integrated into a unique dataset.
Explanations. It is easier for the user to design workflows when explanations are present. Explanations can be on operators for facilitating a design process as well as on results to help the user interpret them. This can be done, for instance, by giving useful instructions about statistical concepts.
Reuse of past experience. Metadata can increase reliability by enabling the reuse of workflows. The reuse of successful cases speeds up the process considerably. It allows to build on prior work and facilitates deeper analysis. It can enable truly collaborative knowledge discovery.
Automatic workflow generation. Metadata can drive the automatic composition and execution of the pre-processing and mining steps. This is the most advanced type of user support but at the same time the most challenging one.
Business understanding. Metadata can provide information about the meaning of the data, the terminology and business concepts and their relationships to the data. Metadata can provide information about the source of the data (provenance) and the path followed from a source to the current site (lineage).

2.3.2 Types of Metadata

The main objects participating in a KDD process include: (1) a *dataset* that needs to be analyzed, (2) *operators* used for pre-processing, and mining, as well as (3) *workflows*, which are combinations of operators with data in the form of directed acyclic graphs. In order to effectively support the user during the analysis, metadata should be stored for every aforementioned object. In addition, metadata that can boost the user support and which were not considered in this context are (4) *domain knowledge* used to store information for the concrete domain of data and (5) *lineage metadata*, relevant to justify the results of an analysis.

Metadata on the input dataset. The idea of characterizing a dataset has been researched from the early inception of meta learning. A dataset that needs to be analyzed - containing all the attributes that are relevant to the problem at hand - is assumed to be selected in advance and is generally described by the following groups of characteristics:

- *General measures:* include general information related to the dataset at hand. To a certain extent they are conceived to measure the complexity of the underlying problem. Some of them are: the number of instances, number of attributes, dataset dimensionality, ratio of missing values, etc.

- *Statistical and information-theoretic measures:* describe attribute statistics and class distributions of a dataset sample. They include different summary statistics per attribute like mean, standard deviation, etc.

However, if the problem to be solved is a prediction problem, then, a variable (or more) is defined to be a response variable. Once the response is defined, further metadata measuring the association between the remaining (input) variables and the response(s) (output) can be used to describe the dataset. Hence, we can additionally have the following groups of dataset characteristics:

- *Geometrical and topological measures:* this group tries to capture geometrical and topological complexity of class boundaries [41]. It includes non-linearity, volume of overlap region, max. Fisher's discriminant ratio, fraction of instance on class boundary, ratio of avg. intra/inter class nearest neighbour distance, etc.

- *Landmarking and model-based measures:* this group is related to measures asserted with fast machine learning algorithms, so called *landmarkers*, and its derivative based on the learned models. It includes error rates and pairwise $1 - p$ values obtained by landmarkers such as 1NN or DecisionStump as well as histogram weights learned by Relief or Support Vector Machines (SVM).

Metadata on Operators. They are typically expressed in the form of semantic information (e.g., ontology). By operators we mean all the different elements that can operate on a dataset. These include: (1) different transformation methods like normalization, discretization, etc., which are considered to be

pre-processing operators and (2) different kinds of learning algorithms like decision trees, support vector machines, etc., which are considered to be *data mining operators*. Metadata on operators can be *internal* or *external* [82]. External metadata treat an operator as a black-box, which means they only consider metadata with regard to the Input, Output, and some other properties like Preconditions and Effects (IOPE). Internal metadata tear up the box by considering metadata linked to an operator's internal *structure* (e.g., parameters or model type) or *performance* (e.g., speed, accuracy, model complexity).

Metadata on Workflows. The previously mentioned metadata are what systems need in order to provide assistance in terms of constructing valid workflows (e.g., all preconditions or input constraints of algorithms are met). However, the generated workflows may not necessarily be optimal. Moreover, the number of generated workflows can reach thousands, given the vast number of available data mining operators (e.g., Rapidminer, Weka). Thus, there needs to be a way of ranking the workflows. One way to do this is to keep track of metadata about workflows. In the eIDA system for instance, in order to characterize workflows, they follow a process mining-like approach. They extract generalized, relational, frequent patterns over the tree representations of the workflows [70].

Domain Knowledge. The effectiveness and need for domain knowledge in knowledge discovery has been confirmed in past research efforts. It is recognized by [55] that there is a role for domain knowledge in all stages of a KDD process. They demonstrate through examples how the domain expert is needed to (1) help define the problem by, e.g., giving business rules on what a failed transaction is or what is considered a problematic customer (2) assist in the creation of the target dataset by, e.g., defining the structure of the data and the semantic value of the data attribute values. However, in order to make use of it, domain knowledge should be represented by models that computers can understand. Ontologies are some of the successful knowledge engineering advances that can be used to build and use domain knowledge in a formal way. An ontology is an explicit specification of a conceptualization. Normally, it is developed to specify a particular domain (e.g., genetics). Such an ontology, often known as a domain ontology, formally specifies the concepts and relationships in that domain. Note that domain knowledge only partially appears in some IDAs in the form of expert rules, and it is mainly with respect to the algorithms, so it can be alternatively called as *expert knowledge*. Yet, domain knowledge with respect to the data itself is not

used by any of the IDAs in the literature.

Lineage Metadata. The KDD process can benefit from lineage metadata. Lineage metadata is composed of steps used to derive a particular dataset. It can be thought of as a recipe for creating data. The quality of the data for the user's analysis can be evaluated through the lineage of the dataset. Data quality of the source is important because errors introduced tend to inflate as the data propagates. This issue is even more critical when using raw data available in data lakes. The level of detail included in the lineage determines the extent to which the quality of the data can be assessed. If semantic knowledge of the pedigree is available, it is possible to automatically evaluate it based on quality metrics [83]. All in all, lineage metadata can be used to understand and justify the results obtained during the analysis. This kind of metadata is also not considered in IDAs.

2.3.3 Comparison of Metadata in IDAs

In Table 2.1, we show the types of metadata used by IDAs and the types of the provided user support. For each cell in the table we put sign '+' if the system supports the particular concept described in the column and sign '-' if not. From the given table, we identify that many support limitations can be explained with the lack of proper metadata. Moreover, note that systems do not deal with the problem of variable selection (e.g., in a big data environment, provide support in terms of which variables are important to select for the analysis and combine them into a unique dataset) and none of the systems provides support in terms of business understanding. These limitations are due to the lack of appropriate metadata. We believe that domain knowledge and lineage metadata could improve the systems in this regard.

Furthermore, from Table 2.1 and from IDAs in general we can conclude the following:

- ES do not use external metadata on operators (e.g., IOPE), therefore are not able to construct entire workflows.
- MLS use huge number of input metadata but they do not provide support for automatically combining multiple steps.
- CBR similarly to MLS rely on historical data and therefore cannot provide useful support when new cases, non-similar to the historical cases appear.

– PDAS generate automatic workflows but they start from scratch every time. They do not make use of the experience from previous data analysis.

– WCEs allow to construct workflows but they do not provide much guidance.

Table 2.1: Type and role of metadata in IDAs.

Category	System	Metadata Type						Metadata Role						
		Input	Operator		Workflow	Domain Know.	Lineage	Single step supp.	Multistep supp.	Variable selection	Explanational	Reuse	Automation	Business underst.
			Int.	Ext.										
ES	SPRINGEX [75]	+	-	-	-	-	-	+	-	-	+	-	-	-
	MLT Consul. [84]	+	+	-	-	-	-	+	-	-	+	-	-	-
MLS	DMA [36]	+	+	-	-	-	-	+	-	-	-	-	-	-
	NOEMON [48]	+	+	-	-	-	-	+	-	-	-	-	-	-
CBR	CITRUS [26]	+	+	-	+	-	-	-	+	-	+	+	+	-
	AST [61]	+	+	-	-	-	-	+	-	-	-	+	+	-
	MiningMart [68]	+	+	-	-	-	-	-	-	-	-	+	-	-
PDAS	RDM [95]	+	+	+	+	-	-	-	+	-	-	-	-	-
	KDDVM [24]	+	+	+	+	-	-	-	+	-	-	-	+	-
	eIDA [52]	+	+	+	+	-	-	+	+	-	-	+	+	-
WCE	IBM SPSS	+	-	+	-	-	-	-	-	-	+	-	-	-
	SAS	+	-	+	-	-	-	-	-	-	+	-	-	-
	RapidMiner	+	-	+	-	-	-	-	-	-	+	-	-	-
	Weka	+	-	+	-	-	-	-	-	-	+	-	-	-

2.4 Metadata Classification

The analysis in Section 4 showed that IDAs rely heavily on metadata in order to provide user support. In order to classify the identified metadata, we decided to extend the classification provided in [30] and later extended in [93]. Our classification can now capture the whole range of metadata required for the KDD process.

The classification tree is given in Figure 2.1. Note that the shaded shapes belong to the original classifications that consist of the following metadata categories: *Definitional*, *Data quality*, *Navigational*, *Lineage*, and *Ratings*. Each category contains its respective metadata artifacts again denoted as shaded shapes in the figure. Nevertheless, in order to attach the required metadata artifacts, change and extension in the taxonomy was required, note the non shaded shapes. The imposed changes are the following: *Definitional* category

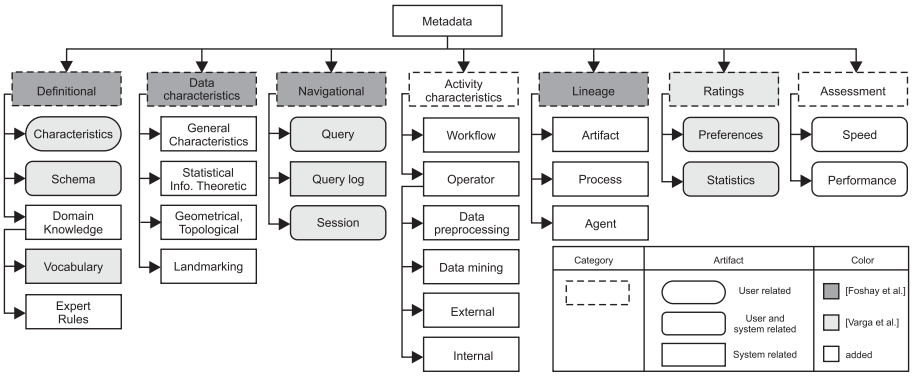


Fig. 2.1: Metadata classification

is extended with a *Domain Knowledge* subcategory which is going to cover metadata related to the domain, *Data quality* is renamed to *Data characteristics* in order to better reflect the meaning of the participating artifacts. An additional category named *Activity characteristics* is added to capture active objects (e.g., operators) in a knowledge discovery process. An additional category *Assessment* is added with the aim of capturing the metadata artifacts with respect to the output of the knowledge discovery process. Next, the *Lineage* category is extended with three metadata artifacts discussed below. Moreover, additional artifacts belonging to different categories are further added. For the purpose of our classification we clearly define all the categories and respective metadata artifacts below. Note however that metadata artifacts that belong to [30, 93] are not discussed extensively. The interested reader is referred to those papers for further information.

The *Definitional* category contains metadata that conveys the meaning of the data to the user or the system. From the original taxonomy in this category there are the integration *schema*, user *characteristics* and a *vocabulary* of business terminology. We extend the *Definitional* category with the *Domain knowledge* subcategory which is going to contain different metadata with regard to the domain. The idea is to enable a knowledge-rich data analysis. However, the goal of a knowledge-rich data analysis is not to provide a priori all the knowledge that might be required but to support a feedback loop by which a small amount of initial knowledge can be bootstrapped into more knowledge by mining, which can in turn be complemented by more human-supplied knowledge to allow further mining, etc. Hence, under the domain knowledge we place the *Vocabulary* artifact from the original classification,

this can be replaced or can easily represent the *domain ontology* discussed in Section 2.3.2. Furthermore, we add *Expert rules* as metadata which can represent some expert knowledge.

Data characteristics consists of artifacts that convey information about the characteristics of data that are of crucial importance to a knowledge discovery process. They advice the system about the completeness or even validity of data. Metadata artifacts in this category are those detected in the analysis in Section 2.3.2.

The *Navigational* category comes from the original classification and keeps track of how the user explores and navigates through data. The metadata artifacts considered under this category can be useful for enabling user support in a data selection phase prior to data mining (e.g., suggesting the user relevant attributes using past experience). Metadata artifacts are: *Query*, *Query log*, and *Sessions*.

The *Activity characteristics* category consists of metadata artifacts whose expressiveness determines the degree of automation that can be achieved in the process of knowledge discovery. These are the most important metadata required in a KDD process. Note that these kind of metadata were not considered in the previous classifications. There are two main metadata artifacts considered here, namely metadata on *Operators* and metadata on *Workflows* (see Section 2.3.2).

Lineage consists of artifacts that model resources (e.g., data-sets) as *Artifacts*, *Processes* (e.g., actions or series of actions performed in artifacts or caused by artifacts, and resulting in new artifacts) and *Agents* (e.g., contextual entities acting as catalysts of a process, enabling, facilitating, controlling, or affecting its execution) [67]. The aim of lineage metadata is to capture the causal dependencies between the artifacts, processes, and agents.

The *Ratings* category comes from the original taxonomy and it contains metadata such as user *Preferences* and usage *Statistics*. However, note that the *Preferences* artifact is important with regard to knowledge discovery as well. It can store different user goals, which can be used by the system to design workflows optimizing some performance measure associated with the user goal. Finally, *Statistics* relates to the data usage indicators. It can keep evidence of which data are explored more.

The *Assessment* category consists of metadata artifacts with regard to the output of a knowledge discovery process. They can be used to assess how good the generated DM workflows are. This is defined by the *Speed* in ex-

ecution and the *Performance* with respect to some evaluation criteria (e.g., predictive accuracy). These metadata can be used to list the best performing workflow or rank all of the constructed workflows.

2.5 Metadata Repository

After having identified the metadata required, we turn on discussing how these metadata can be stored and managed.

The best way to store metadata is to store them in a metadata repository. However, usually metadata remain hidden in scripts and programs, without being further reused. This is also what we realized was happening in practice in a project we developed with a multinational company located in Barcelona[2].

The project aimed at improving the data analytics process in the company. The idea was to allow data analysts to easily select relevant variables for their analysis and assist them during the data pre-processing and mining. The company stores the variables or the data in a raw format in a Data Lake in a Hadoop ecosystem. In order to allow an easy selection of variables and provide user support during the pre-processing phase (e.g., recommend pre-processing operations particularly suited for the domain) we created a semantic repository with the aim of storing all the necessary metadata. The variables in the Data Lake and their respective characteristics are mapped to the corresponding concepts in the repository. In addition, different possible transformations (pre-processing operations; domain knowledge) are described in the repository and they are linked to corresponding concepts. The users are able to easily access the variables through the graphical interface which is fed by the repository. After selecting the variables (e.g., their corresponding concepts) of interest proper transformations are recommended. The information of which pre-processing operators are applied to a given variable are deduced from the metadata repository. Hence, not everybody in the need of analyzing the data needs to be an expert of the domain, as happened to be the case previously in the company. Domain specific knowledge is added once to the repository, and will be used repeatedly by everyone wishing to analyze the data. A high level architecture of the system proposed for the project is shown in Figure 2.2.

The software components accessing the repository are "bound" to the

[2]https://inlab.fib.upc.edu/en/big-data-analytics-lab

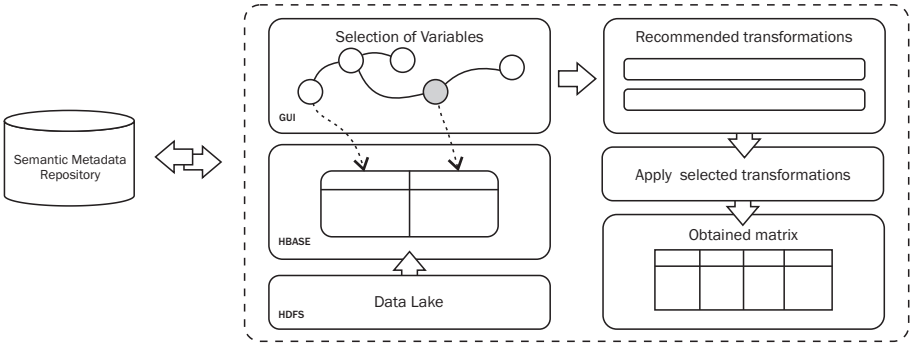


Fig. 2.2: High level view of the proposed system

given metadata structure which is conceptually described by a schema shown in Figure 2.3. The comprehensive schema proposed in this chapter proved to be useful in the project.

The schema can be logically divided into three main parts. The first keeps track of the domain knowledge, the second manages information with regard to passive elements, and they fall under the *IOObject* class, and the third manages information with regard to active elements and they fall under the *Operator* class.

Implementation. We used Resource Description Framework (RDF) as a data model for storing the metadata. In RDF, statements about resources can be made in the form subject-predicate-object expressions and they are called triples. Hence, our repository is defined as a triple store, where we used OpenLink Virtuoso as a storage engine. The repository is provided as a Web Service and an application for metadata management is built on top of it. JavaServer Pages (JSP), Asynchronous JavaScript (AJAX) and XML are used to implement the application and the graphical user interface.

2.6 Related Work

In [30], a taxonomy of the end-user metadata with respect to data warehousing is given. This taxonomy is further extended in [93], where a metadata framework is provided to support the user assistance activities in the context of next generation BI systems. It provides a technical classification of the metadata artifacts required to enable user assistance in retrieving and exploring the data. The focus is on automating certain user related tasks with respect to queries (e.g., query recommendation). Whereas, we are studying

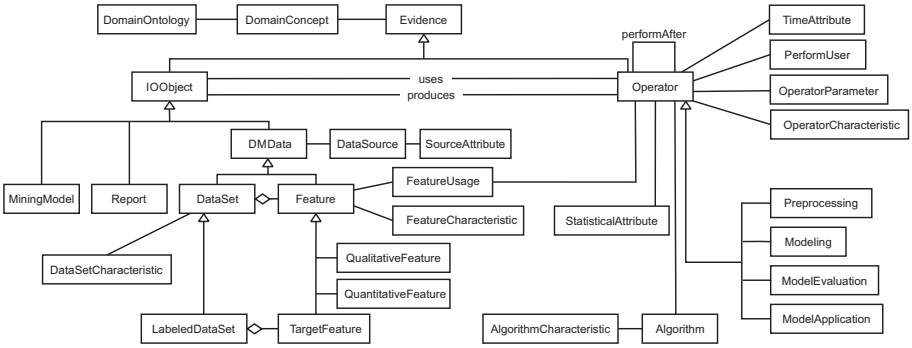


Fig. 2.3: Conceptual schema of the metadata repository

and classifying metadata with the emphasis on how it can help the user during the different steps of KDD.

Another work that can be seen as closely related to us is [82]. The authors provide a comprehensive survey of the systems that make extensive use of metadata to make the automation of knowledge discovery possible. The emphasis is put on explaining the architectures of the systems rather than on a comprehensive classification of metadata.

Finally, Common Warehouse Metamodel [20] provides the necessary abstractions to model generic representations of data mining models, however, the metadata considered does not cover the whole range of KDD steps. It is mainly focused on the metadata for the data mining step. Furthermore, the metadata is considered from the perspective of data interchange, which is how different systems can share and understand metadata with respect to data mining.

2.7 Conclusions

The process of knowledge discovery is challenging. Data relevant to the analysis needs to be selected, pre-processed, mined and finally evaluated. Beginners are alarmed by the myriad of operators and more experienced users limit their activity to several known approaches. A thorough user assistance is necessary. Therefore, systems with the aim of assisting the user during this process are built. We studied these systems with the goal of identifying the metadata used to enable the assistance. Hence, we identified the metadata used to provide user support during the KDD process. We found out that

important metadata such as domain knowledge and lineage which can facilitate data analysts have not been considered. We provided a classification of the metadata found. We proposed a comprehensive metadata framework that captures the complete range of metadata needed to assist the user during the whole process of KDD. We showed the importance of such metadata in a real project by implementing a metadata repository to store and manage the whole range of metadata.

3

On the Predictive Power of Meta-features in OpenML

The demand for performing data analysis is steadily rising. As a consequence, people of different profiles (i.e., non experienced users) have started to analyze their data. However, this is challenging for them. A key step that poses difficulties and determines the success of the analysis is the data mining step (model/algorithm selection problem). Meta-learning is a technique used for assisting non-expert users in this step. The effectiveness of meta-learning, is however, largely dependent on the description/characterization of datasets (i.e., meta-features used for meta-learning). There is need for improving the effectiveness of meta-learning by identifying and designing more predictive meta-features. In this chapter, we use a method from Exploratory Factor Analysis to study the predictive power of different meta-features collected in OpenML, which is a collaborative machine learning platform that is designed to store and organize metadata about datasets, data mining algorithms, models and their evaluations. We first use the method to extract latent features, which are abstract concepts that group together meta-features with common characteristics. Then, we study and visualize the relationship of the latent-features with 3 different performance measures of 4 classification algorithms on hundreds of datasets available in OpenML, and we select the latent-features with the highest predictive power. Finally, we use the selected latent-features to perform meta-learning and we show that our method

improves the meta-learning process. Furthermore, we design an easy to use application for retrieving different metadata from OpenML as the biggest source of data in this domain.

3.1 Introduction

Recently, more and more non-experts are using data mining tools to perform data analysis. These users require off the shelf solutions that will assist them throughout the process. The process known as knowledge discovery or data analytics is shown in Figure 1.1.

One of the key steps of the whole process is the data mining step. A large number of alternative algorithms can be used in this step. Thus, non-experienced users become overwhelmed and require support (e.g., to be recommended the algorithm to use). Various techniques have emerged [82] to provide the required support. Among them, one that has been on the focus of research for long, is *meta-learning* [8, 11, 59].

As we will show next, in short, meta-learning is a process that seeks to find the performance of an algorithm on a given dataset. The ability of predicting the performance of different data mining algorithms allows one to rank the algorithms and therefore provide user support in data mining. However, the success of meta-learning depends on many factors. One of the most important factors is the set of *meta-features* used for meta-learning. Recall that there are two main ingredients in meta-learning: 1) the *dataset characteristics* or the *meta features* plus the performance measure of algorithms on datasets or the *meta response* — these together define the *metadata*, and 2) the *meta-learner*. Yet, the primary source of determining the success of meta-learning, are the chosen *meta-features* (dataset characteristics).

In this chapter, we provide a method for first extracting latent-features, which basically group together meta-features with "common characteristics". Given the latent-features, next, we study their correlation to the performance measure that needs to be predicted. The reason to study the correlation at the latent feature level rather than the meta-feature level, is that first, there is no complete list of meta-features — so any analysis would lack completeness, and second, even if there was such a list, the list would be so large that a sound analysis would not be feasible. Another possibility would be to study the relationship between a coarser group of meta-features and the different performance measures, and make generalizations out of that. That is, for

instance in [79] they study the relationship of the groups of meta-features (e.g., *statistical* meta-features) with the performance measures. We believe this kind of analysis hides the diversely significant relationships of different subsets of meta-features within the same coarse group (i.e., statistical meta-features). That is to say, some statistical features within the same group may have more significant relationships with the performance measure rather than some others in the same group. The analysis at this level, overlooks this and hence the generalizations that all statistical features behave similarly may be incorrect. That is why in this study we settle on a middle ground, where we neither study the individual meta-features nor the groups of such coarse granularity. We make the study in the latent feature level.

There have been many studies with regard to the use (what kinds of metadata to be used) [14], and selection (which are the most relevant) [47] of meta-features or metadata [9] in general. However, these studies have been performed independently and in specific domains. As a matter of fact, the amount of datasets and metadata studied has been relatively small. With the appearance of OpenML [92], however, the idea of collecting and generating metadata and experiments has broadened. OpenML, engages the whole machine learning community to the idea of collecting experiments, datasets and metadata. As a matter of fact, the amount of available data and metadata has naturally increased and is steadily increasing day by day. That is why, our analysis is performed on datasets and metadata provided by OpenML — as the biggest source of data for meta-learning. Specifically, our focus of analysis is on classification problems.

Contributions. The main contributions of this chapter can be summarized as follows:

- We use a traditional method of feature extraction and selection for a novel purpose of studying the predictive power of meta-features in a meta-learning scenario.
- We hand-craft the latent features behind the OpenML meta-features, then we study and visualize their predictive power for predicting different performance measures of different classification algorithms. As a novelty and in contrast to other works, we perform our analysis by splitting the datasets according to the meta-features that can be retrieved from them.
- We evaluate the effectiveness of the method for feature extraction and

selection by performing meta-learning on top of OpenML data, and we show the benefits obtained.

- We develop a user-friendly tool that can be used (e.g., by data analysts) to generate meta-datasets (used for meta-learning) out of OpenML.

The rest of the chapter is organized as follows: In Section 3.2, an overview of meta-learning is given. The method for studying and visualizing the predictive power of meta-features is formally defined and explained in Section 3.3. In Section 3.4, we give an overview of OpenML and more importantly we report on the results obtained after performing our method on top of OpenML. The related work is discussed in Section 3.5. Finally, Section 3.6 concludes the chapter.

3.2 Meta-learning

Most of the data analysis performed remains hidden and not reused. The vast amount of experience gathered from these analysis is not well exploited. The idea behind meta-learning is to exploit the knowledge gained out of this experience. More precisely, meta-learning is the process of learning the relationships between datasets and data mining algorithms. Once different data-mining algorithms have been applied on different datasets, the idea is to use that knowledge in order to assist when data mining algorithms need to be applied on new datasets. As depicted in Figure 3.1, meta-learning consists of 3 steps:

First, a meta learning space is established using metadata consisting of dataset characteristics (meta-features) and a performance measure (meta-response) for data mining algorithms on those particular datasets. Then, there comes the meta-learning phase. Here, a predictive meta-model is generated out of the meta-dataset constructed in the first phase. Finally, in the third step when a new dataset arrives, its characteristics are extracted and

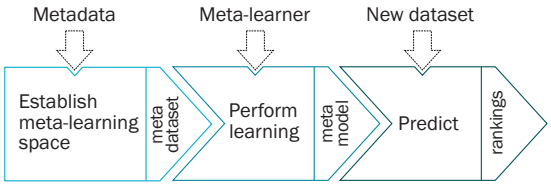


Fig. 3.1: Meta-learning process

the predictive meta-model is used to predict the performance of a particular algorithm — for which the meta-model was built, on that dataset.

This technique can be used to rank different algorithms depending on their predicted performance on a new dataset. Hence, it can be used to recommend data mining algorithms on the data mining step of the analysis.

The two main concepts of meta-learning are the *metadata* and the *meta-learner*. In the following, we briefly discuss these two concepts.

3.2.1 Metadata

Metadata are the necessary information required to establish the meta-dataset. In our definition, they consist of: i) meta-features, and ii) a performance measure of the considered algorithm — meta-response. In statistics, the former are called *predictors* and the latter is called *response*.

Meta-features - characterize a dataset, and initially the two following classes of measures have been proposed:

- *General:* include general information related to the dataset at hand. To a certain extent they are conceived to measure the complexity of the underlying problem. Some of them are: the number of instances, number of attributes, dataset dimensionality, ratio of missing values, etc.
- *Statistical and information-theoretic:* describe attribute statistics and class distributions of a dataset sample. They include different summary statistics per attribute like mean, standard deviation, class entropy, etc.

Since the problem to be solved is usually a prediction problem, and, a variable (or more) is defined to be the response, further meta-features measuring the association between the predictors and the response have been used. These measures are grouped into the *Landmarking and Model-based* class [72, 73]. This class is related to measures asserted with simple machine learning algorithms, so called *landmarkers*, and their derivatives based on the learned models. They include error rates and pairwise $1 - p$ values obtained by landmarkers such as 1NN, DecisionStump or NaiveBayes. Yet, when performed on bigger datasets, these simple machine learning algorithms may introduce significant computational costs.

In order to asses the computational behaviour of the *Landmarking and Model-based* class of measures, we performed an empirical analysis with 720 datasets. In addition to *Landmarking and Model-based*, we calculated the performance of the other classes too, and we show the comparison in terms of

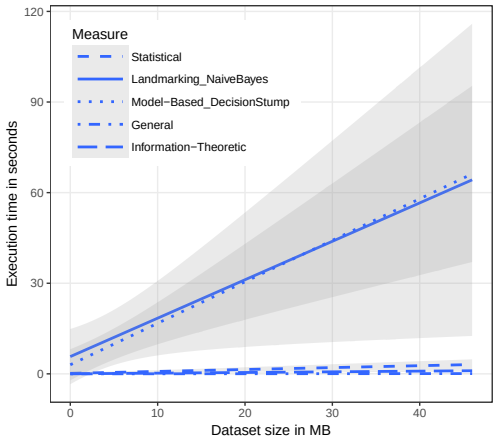


Fig. 3.2: Meta-feature extraction cost. Only *Landmarking and Model-based* classes are represented through single measures. The rest of the classes are represented as the total execution times of their participating measures (cf. Table 3.2)

execution times in Figure 3.2. Mind that the *Lanmdarking and Model-based* class is represented via single measures like the execution time of NaiveBayes and DecisionStump. The other classes are represented through the total execution times of their participating measures. For instance, the execution time of the *Statistical* class is calculated as the total execution time required to retrieve 24 individual measures like *Mean Standard Deviation*, *Mean Skewness*, *Mean Kurtosis*, etc. (cf. Table 3.2). Similarly, the *Information-Theoretic* and *General* class.

For the sake of presentation, instead of showing the scatter plot of the values of all the measures for each dataset, we show the fitted line (regression line) for each measure or class of measures. In addition, the grey areas around the lines denote the 95% interval of the prediction. One can immediately observe the steepness of the slopes, and the wideness of the intervals around the lines representing *Landmarking and Model-based* measures. The former indicates that with an increase in dataset size, retrieving these measures becomes way costlier compared to the rest. The latter indicates that there is a high variability in the execution times for *Landmarking and Model-based* measures. It means that, even if the size of a dataset is small it can still be costly to retrieve them. This can happen for instance if the dataset contains a high number of features and a small number of instances. Hence, as a consequence, because of the way they are computed and their computational overhead we do not consider *Landmarking and Model based* measures as

classical dataset characteristics and they do not participate as meta-features in our experiments.

Performance measures (meta-response) - are different outputs that can be obtained after the evaluation of data mining algorithms. Since we are dealing with classification problems, and hence the algorithms we consider are of classification type, the performance is usually measured in terms of *predictive accuracy*, *precision*, *recall* or the *area under the roc curve (AUC)*. In Table 3.1, formulas for calculating these measures are given. More precisely, classification algorithms are usually evaluated using 10-fold cross-validation [53].

Table 3.1: Performance evaluation measures for classification algorithms

Measure	**Formula**
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Precision	$TP/(TP+FP)$
Recall	$TP/(TP+FN)$
AUC	$Prob(X2 > X1)$

TN - True Negatives; TP - True Positives; FN - False Negatives; FP - False Positives; X1, X2 - Score functions of the classes

3.2.2 Meta-learner

After having generated a meta-dataset with all the necessary metadata, the goal is to build a predictive meta-model that will be able to predict the performance of an algorithm on a new dataset. Formally, the problem can be defined as follows. Given algorithm A and a limited number of training data $D = (\mathbf{x}_1, y_1)...(\mathbf{x}_n, y_n)$, the goal is to find a meta learner with good generalization performance. Generalization performance is estimated by splitting D into disjoint training and validation sets $D^{(i)}_{train}$ and $D^{(i)}_{valid}$. Note that $\mathbf{x} \in x_1, x_2...x_n$ are the dataset characteristics and y is a measure of the performance of algorithm A run on that particular dataset. Hence, $\mathbf{x}$ and y altogether are the extracted metadata. Different meta-learners have been used in the literature, such as, *k-nearest neighbours*, *decision trees*, *support vector machines* [2, 36].

3.3 The Predictive Power of Meta-features: Feature Extraction and Feature Selection

As mentioned above, different and a huge number of meta-features can be used to define the meta-dataset for meta-learning. The list of meta-features may become very big, due to the *meta-meta* effect, referred to as meta^2 [77]. For instance, when a statistical characteristic needs to be taken over the continuous attributes of a given dataset, usually the *mean* of that statistic over the continuous attributes is taken. Let's say, if a general value for *skewness* is in question, commonly the *mean skewness* of all the continuous attributes is considered. However, this does not always need to be the case. One may opt for another statistic, different from the *mean*, for instance, the *skewness* of the *skewness* of continuous attributes or even another, the *kurtosis* of the *skewness* of continuous attributes. As a result, the plethora of statistics that can be computed on top of other statistics, may lead to an explosion of the number of meta-features that can be considered in a meta-learning process.

Therefore, the problem of choosing the most relevant and "minimal" — introduce less computational overhead, set of meta-features is still present. Consequently, the chosen meta-features play a key role in determining the success of meta-learning.

In this section, we discuss the method that we use to study the relevance of the meta-features for predicting the performance measures of data mining algorithms. Our method consists of two steps, which are depicted in Figure 3.3.

In the first step, we perform a Principal Component Analysis (PCA) and subsequently an Orthogonal Rotation on the complete set of meta-features that may be available. Even though this is a standard method in Exploratory Factor Analysis, and has been used in different occasions [65], it is the first time that it is applied in the meta-feature level. Hence, the input required by this step is a meta-dataset consisting of dataset characteristics (meta-features). PCA followed by an Orthogonal Rotation allows us to **extract** the *latent features* and furthermore eliminate the redundant meta-features. Latent features are the abstract features that can be automatically extracted, yet they need to be manually interpreted. The automatic part of the process groups the meta-features with "common characteristics" into latent features. These common characteristics are the abstract concepts that need to be manually interpreted in order to define/describe the latent-features.

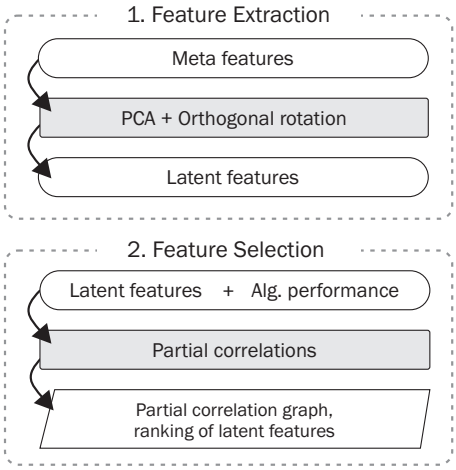


Fig. 3.3: Feature Extraction and Feature Selection

After extracting the latent features we are able to perform the second step. The latent features are used as input for the second step, however, in addition, for all the datasets we compute a performance measure (e.g., predictive accuracy) – *response feature*, of an algorithm (i.e., the one we want to study). Hence, the input of the second step is a meta-dataset, but this time comprised of latent features of datasets and a performance measure of an algorithm run on the respective datasets.

In the second step, we perform a Partial Correlation analysis and generate a *partial correlation graph* that visualizes the relationship between *latent features* and the *response*. This allows us to **select** the *latent features* that are most relevant for predicting the *response*. Hence, at the end of the whole process we obtain a *subset of latent features* — expressed through meta-features, that can be next used for meta-learning. In the following, we briefly and formally introduce the two steps.

3.3.1 Principal Component Analysis

PCA [42] is the predominant linear dimensionality reduction technique, and it has been widely applied on datasets in all scientific domains, from the social sciences and economics, to biology and chemistry. In words, PCA seeks to reduce the dimension of a large number of directly observable features into a smaller set of indirectly observable features — latent features. More precisely, the goals [65] of PCA are, to:

- extract the most important information from the dataset,
- compress the size of the dataset by keeping only this important information,
- explain and simplify the description of the dataset,
- analyze the structure of observations (instances) and variables.

In order to achieve these goals, PCA computes new features, which are called *principal components*. These features are obtained as linear combinations of the original features. The first principle component is required to have the largest possible variance to "explain" the largest part of the variance of the dataset (i.e., meta-dataset). Then, the rest of the components are computed under the following constraints: 1) each component needs to be orthogonal to the previous one, and 2) each component needs to have the largest possible variance. The values of these new features are called *factor scores* and are geometrically interpreted as the *projections* of the instances onto the principal components. These are obtained from the Singular Value Decomposition (SVD) of the dataset X, with:

$$X = P\Delta Q^T \tag{3.1}$$

where P is a $m \times l$ matrix of left singular vectors, Q is the $n \times l$ matrix of right singular vectors and Δ is the diagonal matrix of singular values. l is the rank of the matrix X ($l \leqslant \min\{m, n\}$). The $m \times l$ matrix of factor scores denoted F is obtained as $F = P\Delta$ and can be interpreted as a *projection* matrix because multiplying X by Q gives the values of the *projections* of the observation on the principal components, using Eq. 3.1:

$$XQ = P\Delta Q^T Q = P\Delta = F \tag{3.2}$$

Note that, in Eq. 3.2, matrix F is generated using a standardized dataset — in our case meta-dataset, matrix X with $dim(X) = (m, n)$, where m is the number of instances and n is the the number of features. PCA allows to find a subspace of size p, where the features are grouped depending on their projections into the factor space. The feature groups actually form latent-features/factors. A set of p factors, $p \leqslant n$ is then selected. Each factor represents a certain part of the total variance of the dataset.

3.3.2 Orthogonal Rotation

To facilitate interpretation, after having determined the number of components, the analysis usually involves a rotation of the components retained. Two types of rotations are mainly used: *orthogonal* — the new axes are required to be orthogonal to each other and *oblique* — the new axes are not required to be orthogonal. Note that, the part of variance explained by the total subspace after rotation is the same as it was before the rotation. In this work, orthogonal rotation or more precisely VARIMAX [45] method is chosen to perform a *transformation* of the original data. VARIMAX method assumes that a simple solution means that each component has a small number of large loadings, and a large number of zero loadings. Formally, it searches for a linear combination of the original factors such that the variance of the squared loadings is maximized, which amounts to maximizing v:

$$v = \sum (q_{j,l}^2 - \bar{q}_l^2)^2$$

with $q_{j,l}^2$ being the squared loading of the jth variable of the matrix Q on the component l and $\bar{q}_l^2$ being the mean of the squared loadings. This rotation is performed using the diagonal matrix of singular values and the eigenvectors associated with the correlation matrix of X.

After the rotation, the set of factors — latent features, are more interpretable, and they, of course, are defined by their respective meta-features — the ones that are most correlated with them.

3.3.3 Partial Correlation Graphs

The first step, produces a subset of candidate latent-features for meta-learning. However, it does not provide a relevance measure of the latent-features with respect to the response. So, it does not necessarily retain only the latent-features that are most relevant for predicting the response. The most relevant latent-features with respect to the response out of the ones provided by the first step are derived here, in the second step. That is why, in this step, an additional feature (i.e., response) is attached to the derived set of latent-features. The additional feature can be any of the performance measures of the algorithms (cf. 3.1) evaluated over the datasets — instances of the meta-dataset. Given that, graphical models (i.e., partial correlation graphs) that

represent the relationships between features can be generated[1]. The emphasis of course is on the relationships of the latent-features and the response feature, rather than relationships between latent-features. The latent-features that have very significant relationship with the response are the most relevant ones for predicting the response. Hence, we believe that retaining only the subset of latent-features with very significant relationships with the response is sufficient for performing meta-learning. Furthermore, the set of relevant latent-features may be different for different algorithms considered in a meta-learning framework. More formally, let $\mathbf{x}, \mathbf{y} \in R$ and $\mathbf{z}$ be a random vector. The partial correlation between $\mathbf{x}$ and $\mathbf{y}$, given $\mathbf{z}$ is a measure of association between $\mathbf{x}$ and $\mathbf{y}$ after removing the effect of $\mathbf{z}$. Specifically, $\rho(\mathbf{x}, \mathbf{y}|\mathbf{z})$ is the correlation between $\epsilon_{\mathbf{x}}$ and $\epsilon_{\mathbf{y}}$ where

$$\epsilon_{\mathbf{x}} = \mathbf{x} - \prod_{\mathbf{z}} \mathbf{x}, \quad \epsilon_{\mathbf{y}} = \mathbf{y} - \prod_{\mathbf{z}} \mathbf{y}.$$

Here, $\prod_{\mathbf{z}} \mathbf{x}$ is the projection of $\mathbf{x}$ onto the linear space spanned by $\mathbf{z}$. That is $\prod_{\mathbf{z}} \mathbf{x} = \mathbf{z}\beta$ where β minimizes $\mathbb{E}[\mathbf{x} - \mathbf{z}\beta]^2$. In other words, $\prod_{\mathbf{z}} \mathbf{x}$ is the linear regression of $\mathbf{x}$ on $\mathbf{z}$. Similarly, for $\prod_{\mathbf{z}} \mathbf{y}$.

3.4 Experimental Study on the Predictive Power of OpenML Meta-features

In this section, we first give a brief description of OpenML, then we discuss about the metadata it stores and the application we built to retrieve these metadata in order to generate meta-datasets for meta-learning. After that, we describe the experimental setup which consists of applying the method for feature extraction and selection on OpenML metadata and we discuss the results obtained. Finally, we assess the performance of our method by performing meta-learning on 720 datasets and we show the results obtained.

3.4.1 OpenML

It is an open science platform developed with the aim of allowing researchers to share their datasets, implementations, and experiments (i.e., machine learning and data mining) in such a way that they can easily be found and reused

[1] Taking the partial correlations on the latent features allows to overcome the practical problem of the redundancy that may exist between the meta features, since latent features are "orthogonal".

by others. It offers a web API through which new resources and results can be submitted, and has been integrated into a number of popular machine learning and data mining platforms, such as Weka, RapidMiner, KNIME, and data mining packages in R. They enable an easy and automatic way for submitting new results.

Metadata in OpenML

As previously mentioned, in our definition, metadata consists of meta-features or more precisely dataset characteristics and a response feature or performance measures of different algorithms on datasets. In OpenML, for each uploaded dataset, 61 dataset characteristics[2] (meta features) are calculated. They are listed in Table 3.2. We classify each dataset characteristic into one of the following categories:

- *Continuous* — the dataset characteristic can be calculated only on datasets that contain continuous attributes,
- *Categorical* — the dataset characteristic can be calculated only on datasets that contain categorical attributes,
- *Generic* — the dataset characteristic can be calculated on any dataset.

Classifying meta-features into these three groups is very important, since for instance a *continuous* meta-feature cannot be calculated on datasets with only Categorical attributes and vice-versa, a *categorical* feature cannot be calculated on datasets with only Continuous attributes. Hence, an analysis of meta-features should take this into account. To the best of our knowledge, however, no prior study considers grouping datasets according to the meta-features that can be extracted from them. In [79], they recommend converting the Continuous attributes into Categorical (e.g., by discretization) in order to be able to extract meta-features of *categorical* type in purely Continuous datasets, otherwise the *categorical* features need to be replaced with missing values. However, applying such excessive transformations introduces noise. Thus, in our study (see Section 3.4.2), we group datasets based on their characteristics, and we perform our analysis on these groups separately. That is, the meta-features extracted and considered are in accordance with the types of attributes a dataset contains.

[2]At the time when the study was being performed

Table 3.2: Dataset characteristics in OpenML

No	Name	Type	Class
1	Number of Numeric Attributes	*Continuous*	General
2	Percentage of Numeric Attributes	*Continuous*	General
3..6	Min[Means|Std|Kurtosis|Skewness] of Numeric Attributes	*Continuous*	Statistical
7..10	Mean[Means|Std|Kurtosis|Skewness] of Numeric Attributes	*Continuous*	Statistical
11..14	Max[Means|Std|Kurtosis|Skewness] of Numeric Attributes	*Continuous*	Statistical
15..17	Quartile [1|2|3] of Means of Numeric Attributes	*Continuous*	Statistical
18..20	Quartile [1|2|3] of Std of Numeric Attributes	*Continuous*	Statistical
21..23	Quartile [1|2|3] of Kurtosis of Numeric Attributes	*Continuous*	Statistical
24..26	Quartile [1|2|3] of Skewness of Numeric Attributes	*Continuous*	Statistical
27	Number of Categorical Attributes	*Categorical*	General
28	Number of Binary Attributes	*Categorical*	General
29	Percentage of Categorical Attributes	*Categorical*	General
30	Percentage of Binary Attributes	*Categorical*	General
31..33	[Min|Mean|Max] Attribute Entropy	*Categorical*	Inf. Theo.
34..36	Quartile [1|2|3] Attribute Entropy	*Categorical*	Inf. Theo.
37..39	[Min|Mean|Max] Mutual Information	*Categorical*	Inf. Theo.
40..42	Quartile [1|2|3] Mutual Information	*Categorical*	Inf. Theo.
43	Equivalent Number of Attributes	*Categorical*	Inf. Theo.
44	Noise to Signal Ratio	*Categorical*	Inf. Theo.
45..48	[Min|Mean|Max|Std] Attribute Distinct Values	*Categorical*	Statistical
49	Number of Instances	*Generic*	General
50	Number of Attributes	*Generic*	General
51	Dimensionality	*Generic*	General
52,53	[Number|Percentage] of Missing Values	*Generic*	General
54,55	[Number|Percentage] of Instances with Missing Values	*Generic*	General
56	Number of Classes	*Generic*	General
57	Class Entropy	*Generic*	Inf. Theo.
58,59	[Minority|Majority] Class Size	*Generic*	General
60,61	[Minority|Majority] Class Percentage	*Generic*	General

Regarding the performance measures of algorithms on datasets, all of the performance measures defined in Table 3.1, are stored in OpenML for different classification algorithms. Hence, one can use the metadata provided by OpenML in order to define meta-datasets for meta-learning.

Metadata Retrieval from OpenML

In order to create meta-datasets for meta-learning out of OpenML, one needs to have a good knowledge of the schema of the OpenML repository — the database consists of around 40 tables. This may pose challenges, especially to data analysts with a statistics background. In order to facilitate this process, we first developed a simple application — available for researchers[3], that is capable of generating a meta-dataset for any chosen data mining algorithm. A screenshot of the application is given in Figure 3.4. After generating some meta-datasets we were able to continue with our studies, explained next.

[3]https://github.com/bbilalli/MetadataFromOpenML

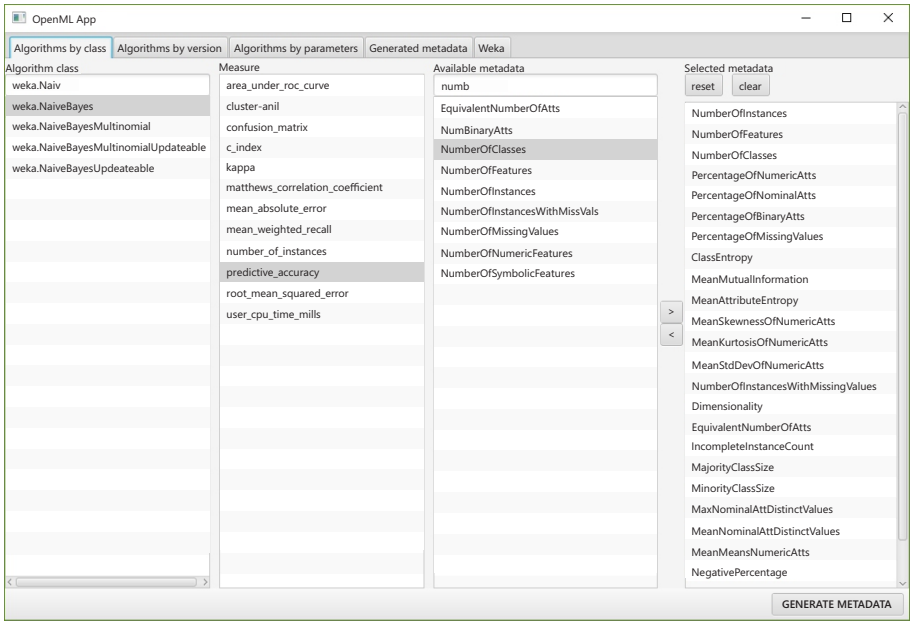


Fig. 3.4: Application for metadata retrieval from OpenML

3.4.2 Experimental Setup

In the following, we discuss our experimental setup, which consists of applying the method shown in Figure 3.3 to metadata retrieved from OpenML.

The Retrieved Metadata

As previously mentioned, since our focus is on classification problems, we retrieved metadata with regard to classification problems only. Hence, we retrieved metadata for 720 datasets (classification problems), whose sizes in terms of features are $n \leqslant 1000$ and instances $m \leqslant 1000000$. The dataset characteristics (meta-features) and performance measures (meta-response) retrieved are listed in Table 3.2 and Table 3.1, respectively. For the sake of experiments, the classification algorithms/techniques that we consider are: *Decision Tree, Naive Bayes, JRip, and Nearest Neighbor*.

PCA and Orthogonal Rotation for OpenML

As described in Eq. 3.2, matrix F is generated using a standardized dataset matrix X with $dim(X) = (m, n)$, where m is the number of observations $(m = 720)$ and n is the number of meta-features $(n = 61)$. A set of p factors (latent-features) is then selected $(p = 14)$. Each factor represents a certain part of the total variance of the meta-dataset. Several methods are used to estimate the correct number of factors to represent the data variance and the feature correlation. One that is commonly used, is to retain all the factors that cumulatively represent a fair amount of the total variance (e.g., 80%). The number of factors needed to explain 80% of the total variance in this case is 14. The 14 selected factors altogether represent 80.69% of the total variance. After determining the number of factors, in order to facilitate the interpretation and more clearly define the latent-features, we perform a rotation (VARIMAX) of the retained factors (components). VARIMAX assumes that each factor has a small number of large loadings, and a large number of zero loadings. Table 3.3, presents the most interesting factor loadings (i.e., with a threshold of ± 0.7) from matrix F for meta-features in the subspace represented by the p first factors $(p = 14)$. Factors are orthogonal and describe the correlation between the features in the original space representation. In the first column of Table 3.3, we show the hand crafted latent concepts that stand behind each one of the factors. They explain the meta-features shown in the second column. Finally, in the third column, factor loadings for the

respective meta-features are shown. Factor Loadings — correlations of meta-features with the latent factors, can range from -1 to 1. Loadings close to -1 or 1 indicate that the factor strongly affects the meta-feature. Loadings close to zero indicate that the factor has a weak affect on the meta-feature.

Table 3.3: Latent features obtained from PCA and VARIMAX

Latent-Feature	**Meta-feature**	**Corr.**
Information of Categorical Attributes	MeanAttributeEntropy	0.853
	MinAttributeEntropy	0.925
	Q1AttributeEntropy	0.920
	Q2AttributeEntropy	0.878
Shape of Numeric Attributes	MeanKurtosisOfNumericAtts	0.933
	MeanSkewnessOfNumericAtts	0.874
	MinKurtosisOfNumericAtts	0.927
	Q1KurtosisOfNumericAtts	0.971
	Q1SkewnessOfNumericAtts	0.755
	Q2KurtosisOfNumericAtts	0.962
	Q2SkewnessOfNumericAtts	0.944
	Q3KurtosisOfNumericAtts	0.969
	Q3SkewnessOfNumericAtts	0.927
Variability of Numeric Attributes	MeanMeansOfNumericAtts	0.987
	MeanStdDevOfNumericAtts	-0.994
	MinMeansOfNumericAtts	0.999
	MaxStdDevOfNumericAtts	-0.999
Min. Variability of Num. Attributes	MinStdDevOfNumericAtts	0.821
	Q1MeansOfNumericAtts	0.719
	Q1StdDevOfNumericAtts	0.887
Modality of Categorical Attributes	MaxNominalAttDistinctValues	0.899
	MeanNominalAttDistinctValues	0.874
	StdvNominalAttDistinctValues	0.942
Number of Instances	NumberOfInstances	0.978
	MajorityClassSize	0.935
	MinorityClassSize	0.843
Missing Values	NumberOfInstancesWithMissVals	0.816

	NumberOfMissingValues	0.866
	PercentageOfInstancesWithMissVals	0.830
	PercentageOfMissingValues	0.701
Dimensionality	NumberOfNumericFeatures	-0.955
	NumberOfFeatures	-0.956
	Dimensionality	-0.770
Information of the Response	ClassEntropy	-0.953
	NumberOfClasses	-0.773
	MajorityClassPercentage	0.770
Number of Cat. Atts.	NumberOfSymbolicFeatures	0.964
	NumberOfBinaryFeatures	0.966
Q3 Lev. Num.Atts.	Q3MeansOfNumericAtts	0.974
	Q3StdDevOfNumericAtts	0.977
Shape of the Extreme Num. Attributes	MinSkewnessOfNumericAtts	0.719
	MaxKurtosisOfNumericAtts	-0.845
	MaxSkewnessOfNumericAtts	-0.794
Mutual Information	MeanMutualInformation	-0.942
	MinMutualInformation	-0.781
	MaxMutualInformation	-0.864
	Q1MutualInformation	-0.803
	Q2MutualInformation	-0.895
	Q3MutualInformation	-0.904
Noise to Signal	EquivalentNumberOfAtts	0.976
	NoiseToSignalRatio	0.975

Partial Correlation Analysis for OpenML Metadata

After having defined the latent features, it is time to rank them according to their relevance for predicting the performance of a given algorithm. In order to physically represent an abstract latent feature we use its corresponding meta-features. That is, a latent feature is physically represented as the average of the meta-features it explains (cf. Table 3.3). Taking the average of meta-features to define the latent concepts, and not their actual weights obtained in the specific instances of the datasets analyzed, gives a more robust

measure of the latent feature, independent of the actual data, which can be easily generalized to future datasets by omitting not existing meta-features or including new ones, provided that they are related with the actual concepts present in the latent features. We join the latent-features obtained, with the performance measures (i.e., accuracy, precision, recall, and AUC) of the algorithms we consider for the corresponding datasets. Now, for each performance measure of every algorithm considered, partial correlation graphs can be generated in order to find the latent-features that are more relevant for predicting the respective performance measures.

However, as previously mentioned, all the datasets retrieved from OpenML do not contain the same types of attributes, and hence all the available meta-features can not be calculated for all of them. Thus, we split the datasets into 3 sets:

- **Combined** - consists of datasets that contain both continuous and categorical attributes (226 datasets) — all meta-features can be calculated (cf. Table 3.2).
- **Continuous** - consists of datasets that do not contain categorical attributes (418 datasets) — only meta-features of type *continuous* and *generic* can be calculated (cf. Table 3.2).
- **Categorical** - consists of datasets that do not contain continuous attributes (76 datasets) — only meta-features of type *categorical* and *generic* can be calculated (cf. Table 3.2).

As a consequence, partial correlation graphs are generated for all the performance measures of all the algorithms for every split of datasets, separately. A study performed this way, distinguishing the groups of datasets, allows one to build a more customized meta-learning system. In addition, our hypothesis is that depending on the datasets, depending on the algorithms, and depending on the performance measures used, the relevance/importance of meta-features differs.

3.4.3 Experimental Results

In this section, we first discuss the results obtained after applying the method for feature extraction and selection on four classification algorithms and three different performance measures. Next, we use the extracted/selected features for performing meta-learning and we show the results obtained.

Results on Decision Trees

In Figure 3.5, the partial correlation graphs for decision trees are shown. Figure 3.5a shows the results with respect to Accuracy, 3.5b with respect to Precision and 3.5c with respect to AUC. Note that the graph for Recall is omitted due to the fact that identical values with Accuracy are obtained (see weighted recall in Weka[4]).

In Figure 3.5, shaded nodes represent the different types of meta and latent features. The white nodes represent the response features (performance measures). The presence of an edge from a latent-feature node to a performance measure node indicates that there exists a significant correlation between the respective features. The correlation is considered to be significant, if the $p - value$ of the correlation is $pval \leqslant 0.01$. The thickness of the edge represents the level of significance. Furthermore, dashed edges represent negative correlation and full edges represent positive correlation. Finally, the different shades in the edges are used to denote the set of datasets where the significant correlation appears.

The nodes connected with short straight edges to the latent-feature nodes, represent the meta-features that define the respective latent-features.

In Figure 3.5a, as it can be observed, the latent feature that is most relevant for predicting the Accuracy is the *information of the response*, and it is attached with three edges. Thus, it is important in all the dataset splits we have considered. Furthermore, the edges are thick — the correlations are very significant, and the edges are full — the correlations are positive. This means the higher the *information of the response* on a given dataset, the higher the Accuracy of a Decision Tree applied on that dataset. The same effect can be observed for Precision (Figure 3.5b), although the edges appear slightly thinner. Hence, for predicting the Accuracy or Precision of Decision Trees, the *information of the response* is very important. The second most important latent-feature which interestingly enough appears as relevant for predicting all the measures, is *mutual information*. However, the correlation appears very significant only in the Combined split and is positive. Another common feature for all the three measures is also *noise to signal*. It is negatively correlated and appears as significant in the Combined split, meaning that the higher it is, the lower will be any of the measures considered.

The rest of the latent-features appear as less significant and are separately relevant for the given measures.

[4]http://weka.sourceforge.net/doc.stable/weka/classifiers/Evaluation.html

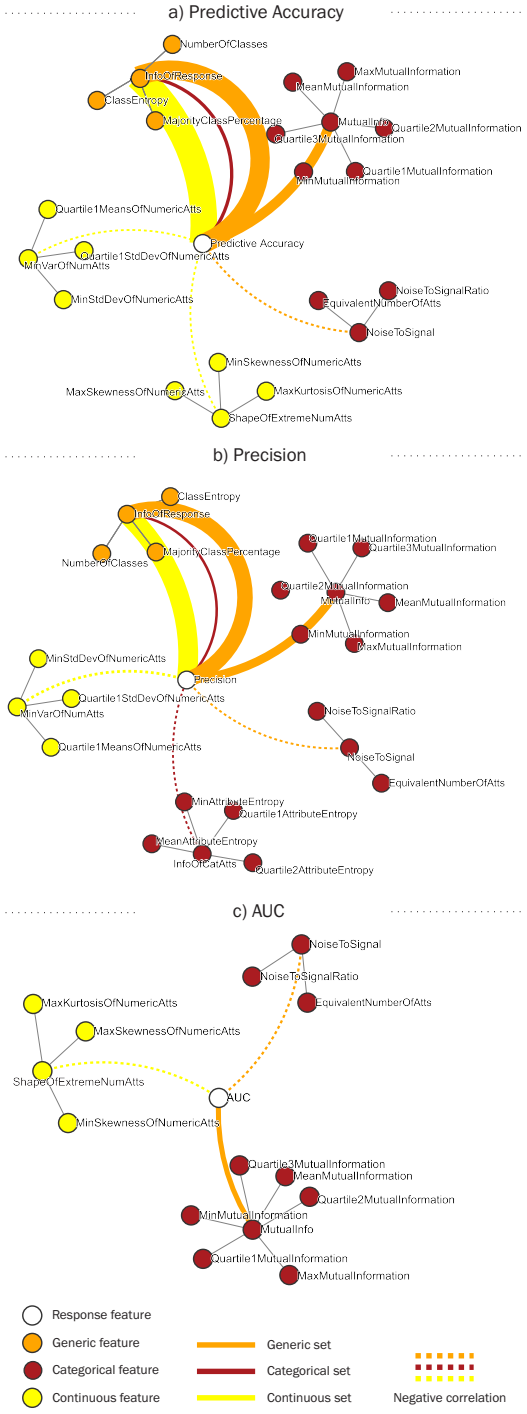


Fig. 3.5: Partial Correlation Graphs for Decision Tree

Table 3.4: The most relevant latent-features

Split	Measure	Alg.	Latent-features: Generic: Info. Of Response	Continuous: Min. Var. Of Num. Atts.	Continuous: Dimensionality	Continuous: Shape Of Extreme Num. Atts.	Categorical: Noise To Signal	Categorical: Info. Of Cat. Atts.	Categorical: Mutual Info.
Comb.	Accuracy	DT	3.5**e-22**				5.0**e-03**		7.1**e-12**
		NB	3.6**e-21**						2.6**e-06**
		JRip	3.2**e-25**						8.2**e-13**
		kNN	5.8**e-17**				6.8**e-04**		2.8**e-09**
	Precision	DT	2.8**e-18**				6.1**e-03**		2.1**e-10**
		NB	1.0**e-24**				5.7**e-03**		2.8**e-07**
		JRip	4.8**e-19**						1.2**e-09**
		kNN	1.1**e-17**				3.6**e-04**		6.9**e-10**
	AUC	DT					4.0**e-03**		4.8**e-06**
		NB					4.2**e-05**		9.4**e-04**
		JRip					5.8**e-03**		2.1**e-04**
		kNN					5.4**e-05**		3.7**e-05**
Cont.	Accuracy	DT	2.6**e-30**	9.2**e-03**		9.7**e-03**			
		NB	1.8**e-18**		1.6**e-03**				
		JRip	3.7**e-35**			9.5**e-03**			
		kNN	4.6**e-18**			6.9**e-03**			
	Precision	DT	3.7**e-23**	7.0**e-04**					
		NB	9.8**e-22**		1.5**e-03**				
		JRip	2.5**e-28**			8.1**e-03**			
		kNN	5.1**e-16**			3.2**e-03**			
	AUC	DT				3.6**e-03**			
		NB			4.3**e-03**				
		JRip				8.4**e-03**			
		kNN				2.8**e-03**			
Cat.	Accuracy	DT	2.0**e-04**						
		NB	3.0**e-06**				2.7**e-03**	1.9**e-03**	
		JRip	5.4**e-05**					5.0**e-03**	
		kNN	8.7**e-04**				1.9**e-05**		
	Precision	DT	1.3**e-03**					7.4**e-03**	
		NB	5.5**e-06**				8.0**e-03**	3.5**e-03**	8.2**e-03**
		JRip	1.4**e-03**						
		kNN	1.8**e-03**				2.3**e-05**		
	AUC	DT							
		NB					1.7**e-04**		
		JRip							
		kNN					3.8**e-05**		

The columns highlighted in gray indicate the latent-features with negative correlation

Results on Naive Bayes, JRip and Nearest Neighbor

Taking into consideration that all the algorithms we consider behave similarly — up to a certain degree, and given that for more algorithms, the graphs may not be very easy to follow, for the rest of algorithms we show the results in a concise table, namely Table 3.4. For the sake of comparison we also add the results of Decision Tree.

We show the results in terms of the *dataset splits*, *performance measures*, *algorithms*, and *latent-features*. Note that we omit the latent-features that do not appear as significant on any of the algorithms. Thus, the presence of a value for a latent-feature denotes that a significant relationship exists between the latent-feature and the corresponding performance measure for a given algorithm and dataset split. Furthermore, the value itself is the *p-value* of the correlation, which in the graphs was represented through the thickness of the edges. One more thing to consider is that the latent-features highlighted in gray are negatively correlated, for all the rows in the table.

While reading the table, one can immediately observe some patterns. The first is that, independently of the split of datasets and independently of algorithms, Accuracy and Precision behave similarly. AUC instead, depends on the split.

Other visible patterns are that, *mutual information* is the measure that appears as relevant for all the algorithms and for all the measures in the Combined split. On the other hand, *information of response* appears as relevant in all the splits for all the algorithms for Accuracy and Precision. Furthermore, the *shape of extreme numerical attributes*, *dimensionality*, and the *minimum variability of numeric attributes* appear as significant in the Continuous split. This is due to the fact that these latent-features are defined only for *continuous* meta-features. On the other hand, *noise to signal*, *information of categorical attributes*, and *mutual information* appear as relevant in the Categorical split. This, because they are defined only for *categorical* features. Finally, an interesting fact about the Categorical split is that for the AUC measure, in Decision Tree and JRip, no latent-features appear as relevant and for the rest of the algorithms only one latent-feature appears as relevant. In addition to this, observing the significance of the correlations of the AUC measure in all the splits, we can realize that they are usually less significant compared to the other measures. The former and the latter altogether indicate that AUC may be more difficult to predict in comparison to the other measures.

Table 3.5: The RMSE values of the predicted measures for all the splits

Split	Measure	Algorithm	RMSE	
			Feature Extraction/Selection	All Features
Combined	Accuracy	Decision Tree	**0.097**	0.123
		Naive Bayes	**0.109**	0.129
		JRip	**0.095**	0.121
		Nearest Neighbor	**0.115**	0.140
	Precision	Decision Tree	**0.109**	0.130
		Naive Bayes	**0.099**	0.125
		JRip	**0.108**	0.129
		Nearest Neighbor	**0.115**	0.138
	AUC	Decision Tree	**0.125**	**0.125**
		Naive Bayes	**0.098**	0.104
		JRip	**0.120**	0.124
		Nearest Neighbor	**0.109**	0.119
Continuous	Accuracy	Decision Tree	**0.091**	0.102
		Naive Bayes	**0.108**	0.122
		JRip	**0.090**	0.103
		Nearest Neighbor	**0.118**	0.130
	Precision	Decision Tree	**0.100**	0.108
		Naive Bayes	**0.099**	0.114
		JRip	**0.094**	0.102
		Nearest Neighbor	**0.112**	0.123
	AUC	Decision Tree	**0.106**	0.107
		Naive Bayes	**0.103**	0.106
		JRip	**0.110**	0.118
		Nearest Neighbor	**0.123**	0.127
Categorical	Accuracy	Decision Tree	0.138	**0.136**
		Naive Bayes	**0.119**	0.128
		JRip	0.132	**0.131**
		Nearest Neighbor	0.151	**0.144**
	Precision	Decision Tree	0.153	**0.148**
		Naive Bayes	**0.126**	**0.126**
		JRip	0.150	**0.145**
		Nearest Neighbor	0.155	**0.149**
	AUC	Decision Tree	0.168	**0.160**
		Naive Bayes	**0.115**	0.123
		JRip	0.164	**0.156**
		Nearest Neighbor	0.181	**0.165**

Bold numbers indicate the best results per row

Results on Meta-learning

The method for studying the predictive power of meta-features, allows to define latent-features and then by measuring their relevance for predicting a performance measure it helps on selecting the most relevant ones. Therefore, at the end, only a subset of features is retained. In terms of meta-learning, using a subset of features instead of the complete set (i.e., 61), has many benefits. First, it saves computational effort when meta-features need to be retrieved. Second, less features means less computational time when building models (i.e., meta-models). Finally, since the retained features are latent-features, the models built on top of them are more generalizable. Yet, in order to enjoy these benefits, the models built with the selected features need to perform well. That is, they need to be as good as, or even better than the models built using all the meta-features. In order to check for this, we trained a regression model (i.e., meta-model) on top of 720 datasets using *Random Forest* as a meta-learner. We evaluated the performance of the regression model using leave-one-out cross-validation. For measuring the performance of the meta-learner we used the Root Mean Squared Error (RMSE), which is often used as a measure of precision and can also serve as a confidence indicator for the predictions.

In Table 3.5, column *Feature Extraction/Selection* shows the RMSE values

Table 3.6: The RMSE values for the Categorical split

Split	Measure	Alg.	RMSE		
			Feature Extraction/Selection	Feature Extraction	All Features
Categorical	Accuracy	Decision Tree	0.138	**0.131**	0.136
		JRip	0.132	**0.126**	0.131
		Nearest Neighbor	0.151	**0.139**	0.144
	Precision	Decision Tree	0.153	**0.147**	0.148
		JRip	0.150	**0.141**	0.145
		Nearest Neighbor	0.155	**0.142**	0.149
	AUC	Decision Tree	0.168	**0.154**	0.160
		JRip	0.164	**0.149**	0.156
		Nearest Neighbor	0.181	**0.165**	**0.165**

Bold numbers indicate the best results per row

of the trained regression models for the sets of features suggested — the union of the relevant latent-features per split is taken, and column *All Features* shows the RMSE values for the complete set of meta-features. Furthermore, the results are classified in terms of the *splits of the datasets*, the *measures*, and the *classification algorithms* used. It can be observed that meta-learning with the feature extraction/selection method applied, performs better than when all the features are used. The only exceptions are in the Categorical split. We believe this is due to that fact that latent-features that appear as relevant in the Categorical split are less significant compared to the ones in the rest of the splits — they are in the order of 10^{-4} (cf. Table 3.4). This furthermore is due to the fact that the Categorical split consists of only 76 datasets — significance values are affected by the sample size.

Note that we can think of significance as a measure of confidence. That is, the more significant the correlation, the more confident you can be about the predictive power of the feature in consideration. Hence, since the correlations in the Categorical split are less significant, we cannot be confident that the selected features are exactly the ones with more predictive power. In order to remedy this problem, we repeated the evaluation for the problematic cases of the Categorical split, however this time using all the extracted latent-features. The results are shown in column *Feature Extraction* in Table 3.6 — we keep the previous results too, for the sake of comparison. It can be observed that using all the extracted latent-features the results improve, and they furthermore become better than when using all the meta-features. This indicates that the extracted latent-features are more robust than the original meta-features. It also indicates that when faced with small sample sizes (i.e., small number of datasets), we can opt for using only the first step of the method depicted in Figure 3.3 (i.e., only Feature Extraction).

3.5 Related Work

Meta-feature definition. Most of the focus with regards to the metadata in meta-learning has been on defining different dataset characteristics that can be used in meta-learning. The first attempt to characterize datasets was done by Rendell [80]. Yet the description of a dataset in terms of its information/statistical measures for the first time appears within the framework of STATLOG project [63], 15 dataset characteristics have been used. This set of characteristics has been later used in various studies for solving the

algorithm selection problem [10, 91]. Sohn [85], notices that some of the characteristics are highly correlated, and she omits the redundant ones in her study. In [14], authors provide formulas for different data characteristics (meta-features) and theoretically discuss their relevance. However, their assumptions are based on intuition and theoretical knowledge. As a consequence, the conclusions are more generic.

An alternative approach to characterize datasets called landmarking has been proposed in [3, 73]. The intuitive idea behind landmarking is that the performance of a simple learner, called landmarker, can be used to predict the performance of given candidate algorithms. Landmarking measures have been evaluated and have shown to perform well in many works, including [5, 33, 73, 79]. The usefulness of these measures, comes with a price though, which is the computational cost. Yet, none of the studies apart from [5], properly acknowledge this fact. This is mainly because the studies are performed on a small number of datasets and on datasets of small sizes. For instance in [79], they perform studies on top of 54 datasets. Another group of measures, quite related to landmarking are model-based measures. The idea is to create a model from the data and use its properties as feature values. The used model in this context is typically a decision tree [4, 72]. These measures have been evaluated in [72, 79]. However, similarly to landmarking, they induce computational overhead when computed on datasets of bigger sizes.

Meta-feature selection. The first attempt at meta-feature selection appeared in the meta-learning framework of zooming-ranking [90]. In this study, some experiments are shown where a classical feature selection method is applied to select relevant features. In [47], they study the meta-feature selection problem too. However, their method is constrained on finding relevant features for pairs of algorithms. This is because their definition of meta-learning is based on detecting the best classification algorithm in a context of pairs of algorithms. In [79], an empirical evaluation of different categories of meta-features in the context of their suitability for predicting classification accuracies for a number of standard classifiers can be found. In addition, an automatic feature selection method is applied to the complete set of meta-features used. However, no finer details of the feature selection method are given. Furthermore, the number of datasets used is very small.

3.6 Conclusions

In this chapter, we used a method to tackle the problem of meta-feature extraction and selection. The method relies on a rigorous mathematical framework and it is beneficial for improving the success of meta-learning tasks. It consists of first extracting latent-features out of meta-features, and then, by studying and visualizing the relationships of the latent features with the response (i.e., the performance measures of algorithms), it allows to select the most relevant or informative latent-features.

After applying the method to data retrieved from OpenML, we were able to observe that: 1) all latent-features are not similarly relevant for predicting the performance of different classification algorithms, and vice versa for predicting different performance measures for the same classification algorithms, 2) all latent-features are not similarly relevant when meta-learning space consists of datasets with specific types of attributes — splitting the datasets in accordance to the meta-features that can be extracted from them was a novelty compared to previous works, and it played a decisive role in our analysis. E.g., the latent-features relevant in a set of datasets with only continuous attributes are not the same as the latent-features relevant in a set of datasets with only categorical attributes, 3) the method for meta-feature extraction/selection improves the meta-learning process.

Having observed this, we claim that meta-feature extraction/selection is a necessary pre-processing step for meta-learning. Moreover, we contend that meta-learning space needs to be specifically customized taking into consideration the available datasets, algorithms, and the performance measures that need to be predicted in a meta-learning framework.

4

Intelligent Assistance for Data Pre-processing

A data mining algorithm may perform differently on datasets with different characteristics, e.g., it might perform better on a dataset with continuous attributes rather than with categorical attributes, or the other way around. Typically, a dataset needs to be pre-processed before being mined. Taking into account all the possible pre-processing operators, there exists a staggeringly large number of alternatives. As a consequence, non-experienced users become overwhelmed with pre-processing alternatives. In this chapter, we show that the problem can be addressed by automating the pre-processing with the support of meta-learning. To this end, we analyze a wide range of data pre-processing techniques and a set of classification algorithms. For each classification algorithm that we consider and a given dataset, we are able to automatically suggest the transformations that improve the quality of the results of the algorithm on the dataset. Our approach will help non-expert users to more effectively identify the transformations appropriate to their applications, and hence to achieve improved results.

4.1 Introduction

One of the most important steps of the knowledge discovery process (cf. Figure 1.1) is the data pre-processing step. Data pre-processing is so important that usually 50-80% of analysis time is spent on it [69]. One of the reasons for this is that a properly prepared/pre-processed dataset yields better results. One can apply the best learning algorithm, but if the data is not well-prepared, the algorithm may perform poorly (e.g., bad predictive accuracy) [19].

Since data pre-processing is so important and typically it is performed by a non expert-user, there is a need to support the user by means of automating the process as much as possible.

In this chapter, we propose a solution to this problem. We aim at assisting the user by recommending transformations, i.e., pre-processing operators, that will ultimately improve the result of the analysis, that usually happens to be a classification task. In order to do that, we make use of the concept of *meta-learning*, which consists of two phases, such as *learning* and *predicting*. For a given dataset and a selected classification algorithm we are able to suggest transformations that once applied yield an improved classification performance (e.g., predictive accuracy).

Contributions. The main contributions of this chapter can be summarized as follows:

- We leverage ideas from meta-learning to present a technique for ranking pre-processing operators depending on their impact on the final result of data analysis.
- We show the benefits of our approach by implementing a prototype that is capable of automatically recommending pre-processing operators to the user.
- We show experiments that demonstrate the effectiveness and quality of our approach.

The rest of the chapter is organized as follows: an overview of data pre-processing together with its benefits is given in Section 4.2. Our proposed solution is formally defined in Section 4.3. A brief look at the materialization of our proposed approach in terms of a prototype solution is given in Section 4.4. The results of the experimental evaluations are reported in Section 4.5.

The related work is discussed in Section 4.6. Finally, Section 4.7 summarizes our work in this chapter.

4.2 Overview on Data Pre-processing

In this section we give a general overview of the pre-processing step in data analytics by first explaining the different existing pre-processing operators/algorithms. Next, we examine and discuss the impact of data pre-processing operators on the final result of data analysis.

4.2.1 Data Pre-processing Operators

Traditionally, data mining has been performed on transactional data consisting of continuous attributes. The continuous scale of these attributes has enabled the use of conventional statistical methods, such as logistic regression. However, the advances in computational and storage capacity have enabled the accumulation of ordinal, nominal, and binary data, giving rise to datasets of heterogeneous scales. This has induced: 1) advances in the application of data driven methods (e.g., decision trees, bayesian algorithms, nearest neighbours, support vector machines, etc.) capable of mining large datasets, and 2) challenges in transforming attributes of different scales into mathematically feasible and computationally suitable formats [19]. Indeed, each attribute may require special treatment, such as discretization of numerical attributes, rescaling of ordinal attributes, and encoding of categorical ones. Hence, different transformations may be required and the ones we con-

Table 4.1: List of transformations (data pre-processing operators)

Transformation	Technique	Attributes	Input Type	Output Type
Discretization	Supervised	Local	Continuous	Categorical
Discretization	Unsupervised	Local	Continuous	Categorical
Nominal to Binary	Supervised	Global	Categorical	Continuous
Nominal to Binary	Unsupervised	Local	Categorical	Continuous
Normalization	Unsupervised	Global	Continuous	Continuous
Standardization	Unsupervised	Global	Continuous	Continuous
Replace Miss. Val.	Unsupervised	Global	Continuous	Continuous
Replace Miss. Val.	Unsupervised	Global	Categorical	Categorical
Principal Components	Unsupervised	Global	Continuous	Continuous

sider are shown in Table 4.1. They are available in the form of open source packages in different data mining tools (e.g., Weka, RapidMiner). We aimed at selecting some of the most important transformations that cover a wide range of data pre-processing tasks, which are distinguished as *data reduction* and *data projection*. The purpose of *data reduction* is to decrease the size of the dataset (e.g., instances selection or feature selection). The purpose of *data projection* is to alter the representation of the dataset (e.g., mapping continuous values to categories or encoding nominal attributes) [74].

In Table 4.1, a transformation is described in terms of: 1) the *Technique* it uses, which can be `Supervised` — the algorithm knows the class of each instance and `Unsupervised` — the algorithm is not aware of the class, 2) the *Attributes* it uses, which can be `Global` — applied to all compatible attributes, or `Local` — applied individually to specific compatible attributes, 3) the *Input Type*, which denotes the compatible attribute type for a given transformation, which can be `Continuous` — it represents measurements on some continuous scale, or `Categorical` — it represents information about some categorical or discrete characteristics, 4) the *Output Type*, which denotes the type of the attribute after the transformation and it can similarly be `Continuous` or `Categorical`.

4.2.2 Impact of Pre-processing

In the following we devise a brief example that reveals the importance of data pre-processing for a prediction (e.g., classification) problem. For more in depth analysis of the impact of pre-processing we refer the reader to [19, 22].

Table 4.2: Summary of `Automobile`

Metadata	Value
Instances	205
Attributes	26
Classes	2
Categorical Atts.	11
Continuous Atts.	15
Miss. Values	59

Table 4.3: The impact of transformations on the `Automobile` dataset

Transformation	Attribute	PA
Unsup. Discretiz.	1,9,10,11,12,13	0.81
Unsup. Discretiz.	1,9,10	0.80
Unsup. Discretiz.	All Cont. Atts.	0.75
Sup. Nom. To Bin.	All Cat. Atts.	0.73
Unsup. Normaliz.	All Cont. Atts.	0.71

Let us suppose that a user wants to apply the `Logistic` algorithm to the

`Automobile`[1] dataset. The summary of `Automobile` is given in Table 4.2. This dataset specifies *autos* in terms of their various characteristics like *fuel type*, *aspiration*, *num-of-doors*, *engine-size*, etc. The response attribute (i.e., class) is *symboling*. *Symboling* is a categorical attribute that indicates the insurance risk rate, and its range is: `-3,-2,-1,0,1,2,3`. Value `3` indicates that the auto is risky, `-3` that it is pretty safe. The problem is to build a model that will predict the insurance risk rate for a new auto.

Now, if `Logistic Regression` is applied to the original non-transformed dataset, a predictive accuracy of `0.71` is obtained with 10 fold cross-validation. Note that for this run the Weka implementation of `Logistic Regression` with a default parametrization is used. On the other hand, if some pre-processing is first performed on `Automobile` and then the data mining algorithm is applied, the results shown in Table 4.3 are obtained. In Table 4.3, the first column denotes the transformation applied, the second denotes the index values of the attributes to which the transformation is applied and the third is the predictive accuracy (PA) obtained after the `Logistic` algorithm is applied on the transformed dataset. Note that for instance, if the transformation `Unsupervised Discretization` (with default parametrization) is applied to attributes `{1,9,10,11,12,13}`, an improvement of 14% is obtained in terms of the predictive accuracy. A non-experienced user would not be aware of that. Hence, a proper recommendation of transformations would ease user's task and at the same time it would improve the final result.

Indeed, to alleviate this problem, in the next section we propose an approach that uses meta-learning to recommend transformations that ultimately improve the result of the data analysis.

4.3 Meta-learning for Data Pre-processing

Meta-learning is a general process used for predicting the performance (e.g., predictive accuracy) of an algorithm on a given dataset. It is a method that aims at finding relationships between dataset characteristics and data mining algorithms [11].

However, taking into consideration the above mentioned scenario where a user needs to be provided with some transformations to be applied, we propose to use meta-learning in order to find relationships between transformations and data mining algorithms.

[1] https://archive.ics.uci.edu/ml/support/Automobile

This can be done, since transformations, through the changes they cause in the dataset characteristics, impact the results of the data mining algorithms. Using meta-learning, we can *learn* this impact and we can **rank** transformations according to their capability of improving the final result of the data mining algorithm.

The process of ranking consists of three phases, see Figure 3.1 (cf. Chapter 3). First, a *meta-learning space* is established using metadata. The metadata consist of dataset characteristics along with some performance measures for data mining algorithms on those particular datasets. Then, the *meta-learning phase* generates a model (i.e., predictive meta-model) which defines the area of competence of the data mining algorithm [48]. Finally, when a transformed dataset (i.e., a transformation was applied on the dataset) arrives, the dataset characteristics are extracted and fed to the predictive meta-model, which predicts the performance of the algorithm on the transformed version of the dataset. At this point, we are able to obtain predictions for different transformed datasets (e.g., different transformations applied to the same dataset). By comparing the obtained predictions for the different transformations, we are able to rank the transformations depending on their predicted impact on the given dataset. This concludes the *prediction* phase.

For the sake of concreteness, let us assume that, the user wants to apply Logistic Regression to a dataset, to deal with a classification problem at hand. Our system, first, takes the dataset and applies several transformations to it (i.e., one at a time to avoid a combinatorial problem). As a result, several transformed versions of the dataset are obtained. Next, the system extracts the necessary meta-features (cf. Section 4.3.1) from all the transformed versions of the dataset and uses them as input to the predictive meta-model which is specifically built for the Logistic Regression algorithm. The meta-model is built by training a meta-learner (e.g., Random Forest or any other regression algorithm) on existing/historical metadata consisting of dataset characteristics and a performance measure (e.g., predictive accuracy) of Logistic Regression on the datasets. This meta-model is used to produce a prediction for each transformed dataset. Informally, this is what the system thinks will be the result (i.e., predictive accuracy) of applying Logistic Regression on each transformed dataset. Thus, these values are used to rank the transformations. That is, the higher the prediction, the higher will stand the transformation — that caused this prediction, in the ranking. The top ranked transformations will be recommended to the user.

Two necessary ingredients for performing the aforementioned process are the **metadata** and the **meta-learner**. In the following we give details on each one of them.

4.3.1 Metadata

In Chapter 2, we studied and classified all types of metadata that can be used by systems that intelligently support the user during the process of data analysis. These systems may vary in terms of the methodology they follow (e.g., case based reasoning, planning systems, etc.) [82] and may use different metadata. When it comes to meta-learning however, metadata consist of: 1) dataset characteristics — **meta-features**, and 2) a performance measure for the algorithms considered — **meta-response**. In statistics, the former are called *predictors* and the latter is called *response*.

Meta-features

Meta-features characterize a dataset, and two main classes have been proposed :

- *General measures:* include general information related to the dataset at hand. To a certain extent they are conceived to measure the complexity of the underlying problem. Some of them are: the number of instances, number of attributes, dataset dimensionality, ratio of missing values, etc.
- *Statistical and information-theoretic measures:* describe attribute statistics and class distributions of a dataset sample. They include different summary statistics per attribute like mean, standard deviation, class entropy, etc.

In the literature, other meta-features have also been proposed, such as *Landmarking and model-based* [72, 73] measures. These measures are not classical dataset characteristics, since they involve performing simple data mining algorithms on datasets and then use these as values of the features. We do not consider them as dataset characteristics, and since in big data settings, they may introduce significant computational overhead (cf. Section 3.2.1), they do not participate as meta-features in our experiments. Yet, various systems may use various meta-features for the construction of the meta-space.

Table 4.4: Meta-features (dataset characteristics)

No	Name	Type	Modifiable
1..2	[Number|Percentage] of Continuous Attributes	Continuous	Yes
3..6	Min[Means|Std|Kurtosis|Skewness] of Continuous Att.	Continuous	Yes
7..10	Mean[Means|Std|Kurtosis|Skewness] of Continuous Att.	Continuous	Yes
11..14	Max[Means|Std|Kurtosis|Skewness] of Continuous Att.	Continuous	Yes
15..17	Quartile [1|2|3] of Means of Continuous Attributes	Continuous	Yes
18..20	Quartile [1|2|3] of Std of Continuous Attributes	Continuous	Yes
21..23	Quartile [1|2|3] of Kurtosis of Continuous Attributes	Continuous	Yes
24..26	Quartile [1|2|3] of Skewness of Continuous Attributes	Continuous	Yes
27	Number of Categorical Attributes	Categorical	Yes
28	Number of Binary Attributes	Categorical	Yes
29	Percentage of Categorical Attributes	Categorical	Yes
30	Percentage of Binary Attributes	Categorical	Yes
31..33	[Min|Mean|Max] Attribute Entropy	Categorical	Yes
34..36	Quartile [1|2|3] Attribute Entropy	Categorical	Yes
37..39	[Min|Mean|Max] Mutual Information	Categorical	Yes
40..42	Quartile [1|2|3] Mutual Information	Categorical	Yes
43	Equivalent Number of Attributes	Categorical	Yes
44	Noise to Signal Ratio	Categorical	Yes
45..48	[Min|Mean|Max|Std] Attribute Distinct Values	Categorical	Yes
49	Number of Instances	Generic	Yes
50	Number of Attributes	Generic	Yes
51	Dimensionality	Generic	Yes
52,53	[Number|Percentage] of Missing Values	Generic	Yes
54,55	[Number|Percentage] of Instances with Missing Values	Generic	Yes
56	Number of Classes	Generic	No
57	Class Entropy	Generic	No
58,59	[Minority|Majority] Class Size	Generic	No
60,61	[Minority|Majority] Class Percentage	Generic	No

The meta-features we specifically consider are shown in Table 4.4. These are the set of meta-features extracted in OpenML [92].

Column *Type* in Table 4.4, specifies the type of the meta-feature, and it can be `Continuous` — the meta-feature can be extracted only from datasets that contain attributes of continuous type, `Categorical` — the meta-feature can be extracted only from datasets that contain attributes of categorical type, `Generic` — the meta-feature can be extracted from any dataset, regardless of the types of it's attributes. Furthermore, in Table 4.4, column *Modifiable* indicates whether the meta-features are modifiable through the transformations we use, shown in Table 4.1. If meta-features are not modifiable/transformable, we do not consider them, because they remain constant and they do not reflect the impact of transformations.

Yet, note that, in the set of meta-features considered (excluding the non-modifiable ones), not all the meta-features are independent or non-correlated. In order to remedy this, we perform *feature extraction* and then *feature selection* on the original/initial set of meta-features. The method is depicted in Figure 3.3 and consists of two steps, which are explained next.

Feature Extraction. As previously mentioned, some of the meta-features considered may be very correlated or even redundant (e.g., we calculated the correlation between *Noise to Signal Ratio* and *Equivalent Number of Attributes* on a sample with 570 datasets, and they appeared to be correlated with a Pearson coefficient of 0.85)[2]. As a matter of fact, performing meta-learning on top of correlated meta-features will not lead to a good performance of the meta-learning system. Therefore, in order to remove the dependency and extract the most important information from the meta-features, we first perform a Principal Component Analysis (PCA) [42] to the original set of meta-features. PCA is the predominant linear dimensionality reduction technique, and it has been widely applied on datasets in all scientific domains, from the social sciences and economics, to biology and chemistry. In short, PCA seeks to reduce the dimension of a large number of directly observable features into a smaller set of indirectly observable features — *latent features*. PCA finds a subspace of size p, where the features are clustered depending on their projections into the factor space. The feature clusters actually form *latent-features*. A set of p components $p \leqslant n$ is then selected. Each component represents a certain part of the total variance of the dataset. In this chapter we retain all the components (latent features) that cumulatively represent at

[2]http://www.openml.org/search?type=measure

least 90% of the total variance.

Next, to facilitate interpretation, after having determined the number of components, we perform a rotation of the components retained.

In this chapter, orthogonal rotation or more precisely VARIMAX [45] method is chosen to perform a *transformation* of the data. VARIMAX method assumes that a simple solution means that each component has a small number of large loadings, and a large number of zero loadings.

As a final remark, PCA followed by VARIMAX rotation removes the dependency/correlation between features, however, it does not guarantee that all the latent-features retained are equally relevant for predicting the performance of a data mining algorithm. Thus, in order to retain only the most relevant latent features (i.e., the ones that have higher predictive power), in the second step (cf. Figure 3.3), we perform latent-feature selection, which is explained next.

Feature selection. The first step in Figure 3.3, produces a set of candidate latent-features for meta-learning. However, it does not provide a measure on the relevance of the latent-features. The question is: *"How relevant is a latent-feature for predicting the response?"*. Indeed, we are interested in the subset of latent-features that are the most relevant for predicting the response. In order to find and retain only the most relevant latent-features, in this step, first, an additional feature (i.e., response) is attached to the set of latent-features. The additional feature can be any of the performance measures (e.g., predictive accuracy, cf. Section 4.3.1) of the algorithms evaluated over the datasets (the instances of the meta-dataset).

Next, we calculate the partial correlation [1] between the features. This allows us to generate partial correlation graphs that represent the relationships and the strengths of the relationships between features. Our focus is only on the relationships between the latent-features (extracted in the first step) and the response. That is, we measure how relevant are the latent-features for predicting the response. The graph allows us to visualize only the latent-features that have a direct link with the response. Furthermore, these links have a strength which is measured through the significance value (i.e., *p-value*). We consider as significant only the links with a value lower than 0.05 ($p\text{-value} \leqslant 0.05$). Hence, at the end, only a subset of latent-features is retained. Given the fact that, latent-features are defined through the original meta-features, as explained in the previous section, ultimately this step allows us to retain a subset of the original meta-features. Hence, in the whole process then, we

use only the meta-features identified in this step. A schematic representation of a partial correlation graph is shown in Figure 4.1.

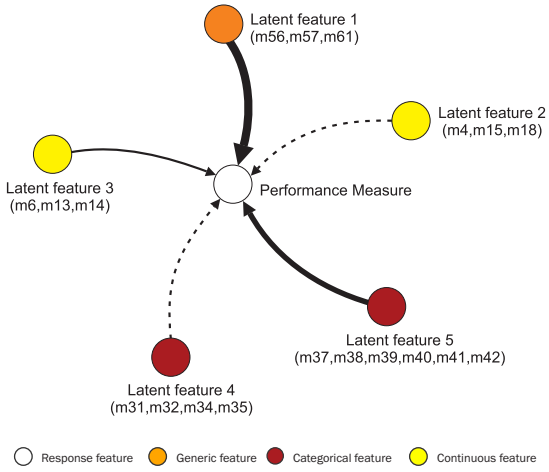


Fig. 4.1: Schematic representation of a partial correlation graph for *Decision Tree* algorithm and *Predictive Accuracy* as a performance measure. The thickness of the edges denotes the significance of the relationship between two nodes. The dashed edges denote negative correlation. m[n] denotes a meta-feature and corresponds to the meta-features in Table 4.4.

Performance measures (meta-response)

Performance measures are different outputs that can be obtained after the evaluation of data mining algorithms. Since we are dealing with classification problems, and hence the algorithms we consider are of classification type, the performance is usually measured in terms of *predictive accuracy*, *precision*, *recall* or *area under the roc curve (AUC)*. Moreover, classification algorithms are usually evaluated using 10-fold cross-validation [53].

In Table 3.1 (cf. Chapter 3), formulas for calculating these measures are given. Briefly, *Accuracy* is a measure of the overall effectiveness of a classifier. *Precision* is the class agreement of the instance labels with the positive labels given by the classifier. *Recall* measures the effectiveness of a classifier to identify positive labels. Finally, one can think of *AUC* as the classifier's ability to avoid false classification. For more details regarding these measures and how they extend to multi-class classification problems we refer the reader to [86].

4.3.2 Meta-learner

Having stored an algorithm performance characteristic (cf. Table 3.1) and a set of dataset characteristics (cf. Table 4.4), the goal is to predict the performance of an algorithm on a transformed dataset. Formally, the problem can be defined as follows. Given algorithm A and a limited number of training data $D = (\mathbf{x}_1, y_1)...(\mathbf{x}_n, y_n)$, the goal is to find a meta learner with optimal/good generalization performance. Generalization performance is estimated by splitting D into disjoint training and validation sets $D^{(i)}_{train}$ and $D^{(i)}_{valid}$. We use leave-one-out validation [53], which splits the training data into n partitions $D^{(1)}_{valid}, ..., D^{(n)}_{valid}$, and sets $D^{(i)}_{train} = D \backslash D^{(i)}_{valid}$ for $i = 1, ..., n$. Note that $\mathbf{x} \in x_1, x_2...x_n$ are the dataset characteristics and y is a chosen measure of the performance of algorithm A run on that particular dataset. Hence, $\mathbf{x}$ and y altogether are the extracted metadata. Since y consists of 4 different performance measures for algorithm runs, we build meta-spaces for each specific measure separately. Then for each meta-space (meta-dataset), we generate meta-models — using a meta-learner.

A few basic criteria were followed for selecting the meta-learner to use. First, the problem in the meta-learning space is of regression type — a number needs to be predicted (i.e., a value in the range of $[0, 1]$) rather than a class.

The second criterion is that the meta-learner needs to be more *sensitive*. By this we mean that the meta-learner needs to be able to capture even the slight changes that transformations might apply on datasets. This is because we need to predict the impact of the transformations on the data mining results and we need to be able to compare the impacts of different transformations. This comparison needs to be done at a finer granularity. Otherwise, in the worst case, all the transformations may end up having the same impact. For instance, as a first approach we considered simple regression trees [7] as meta-learners, and they suffer from this problem. Their limitation is that they contain a discrete number of leaves, and hence a discrete number of possible predictions.

The third criterion is that the meta-learner should handle missing values. Recall that some dataset characteristics can be calculated on datasets that necessarily contain either continuous or categorical attributes (cf. Table 4.4). As a matter of fact, our second trial of using *Logistic Regression* as meta-learner did not give good results, because *Logistic Regression* cannot handle missing values.

Hence, finally the meta-learner we decided to use is *Random Forest*. *Random Forest* complies with all the above mentioned criteria. It can be used for regression problems. It suffers far less from the discreteness of the leaves, because internally, a lot of trees (i.e., 500 trees) are built at random and at the end averages are taken to be used as predictions. Finally, it performs well when missing values are present.

Thus, we use *Random Forest* to build models for each data mining algorithm or more precisely for each classification algorithm that we consider.

In particular, the classification algorithms that we consider are representative algorithms for all, except two classes of algorithms in Weka. In Weka, the classification algorithms are classified into: *bayes, functions, lazy, rules, trees, meta-methods, and miscellaneous*. We aimed at considering one algorithm for each one of the first five classes, and they are: *Naive Bayes, Logistic, IBk, PART, and J48,* respectively. The last two classes were omitted due to the fact that they are more complex and are not commonly used by non experienced users.

4.4 Solution Prototype

In this section, we discuss the materialization of the approach proposed in Section 4.3, into a prototype solution. The general architecture of the developed prototype solution is depicted in Figure 4.2. The solution's main processes, **Learning** and **Recommending**, are implemented independently from each other. Below we give detailed explanations for each one of them.

4.4.1 Learning Phase

In the previous sections we mentioned that in order to build a model (i.e., predictive meta-model), we must firstly establish the meta-space — denoted as *Learning phase* in Figure 4.2. In our context, the meta-space needs to be constructed out of metadata that can be extracted from datasets and from the executions of classification algorithms on those datasets. As a matter of fact, we needed to fetch hundreds of datasets, extract their characteristics, run different algorithms on them and get different evaluation measures with 10 fold cross validation. Finally, use all of these to feed the *Meta-database*.

In order to do the aforementioned, we first used OpenML to fetch several hundred datasets (i.e., 570). Next, from each dataset we extracted the 55 dataset characteristics – highlighted as modifiable in Table 4.4, and on each

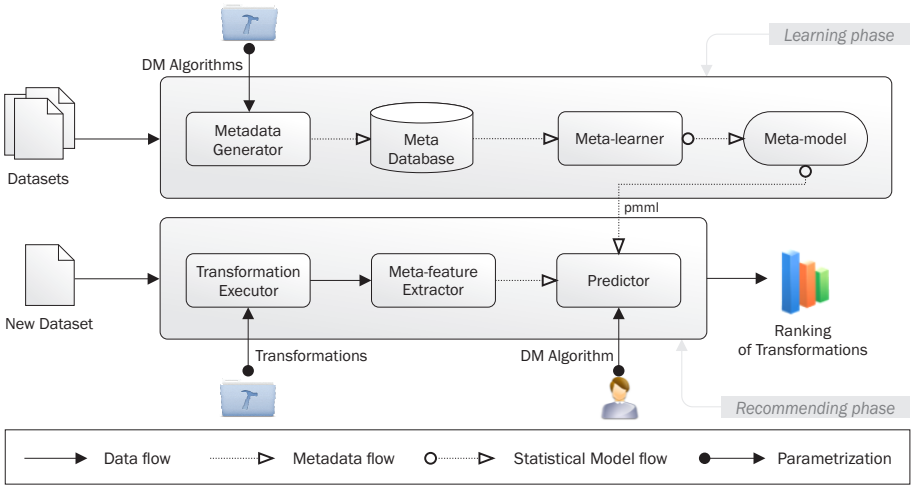


Fig. 4.2: Solution architecture

dataset we applied 5 classification algorithms in order to extract the performance measures — shown in Table 3.1. Then, we performed feature extraction — using PCA followed by VARIMAX on the set of dataset characteristics (meta-features), and feature selection — using the partial correlation graphs, for every classification algorithm and every performance measure considered. Finally, for each classification algorithm and for each performance measure, we obtained a meta-dataset that was fed to the *Meta-database*. In Figure 4.2, this whole process is represented via the *Metadata Generator* module and was developed in `Java`.

After obtaining the metadata, hence constructing the meta-space, we continued on building the *Models* (or predictive meta-models) using the *Meta-learner* (i.e., Random Forest) we considered. We used the `R` language to construct a model for each one of the algorithms and for each one of the performance measures considered. After that, the models were exported to PMML [37] files, and were next fed to the *Predictor* in the recommending phase.

Note that this process is not specifically tailored for datasets from the OpenML repository, but it can work on any collection of datasets. The models obtained are expected to slightly change from one collection to another.

4.4.2 Recommending Phase

When a user wants to analyze a dataset, he/she selects an algorithm to be used for the analysis and then the system automatically recommends transformations to be applied, such that the final result is improved. In order to do that, the system first applies different transformations to the dataset through the *Transformation executor* module. Then, the meta-features of the transformed dataset are extracted through the *Meta-feature Extractor* module and they are fed to the *Predictor*, which using the meta-model (i.e., PMML file) corresponding to the classification algorithm selected by the user, predicts the impact of the transformation/s. The gain here is that, the classification algorithms are not applied for real to the transformed datasets — which is a costly process. Instead, meta-models are used to predict the outputs of the classification algorithms on the transformed datasets. Hence, finally, transformations are ranked according to their predicted impact on the final result — according to whether they improve the final result. The modules of the *Recommending phase* are entirely developed in `Java`.

4.5 Evaluation

We perform an experimental study of the performance that can be achieved by our approach on various algorithms and various datasets. After specifying our experimental environment, we evaluate our system's ability to predict the transformations that improve the final result of the analysis.

4.5.1 Experimental Setup

Recall that when building the meta-learners, we use leave-one-out validation for evaluating them. Likewise, in order to enable a larger number of datasets for performing the experiments, each time we performed the leave-one-out validation, we created a meta-model using the subset of datasets (i.e., withholding the dataset that was left-out). Hence, for each data mining algorithm, we created as many meta-models as datasets considered for the respective algorithm. As a matter of fact, in order to perform experiments for an algorithm, we can use the entire set of datasets for testing, only bearing in mind that for each dataset, in the *Predictor*, we use the meta-model that was built without using that particular dataset.

In this context, an experiment — depicted in Figure 4.3, is performed in

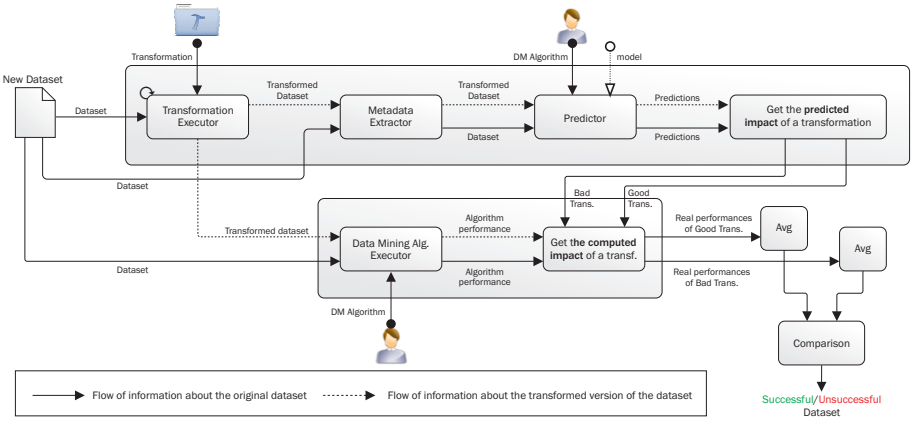


Fig. 4.3: Experimentation scheme

the following way. First, a dataset and a classification algorithm to be used for performing analysis (i.e., classification) on the dataset is selected. Next, the system finds the **impact** of a **set of transformations** on the final result of the classification.

The **set of transformations**, consists of iteratively applying the transformations shown in Table 4.1, however each time changing the set of attributes to which the transformation is applied. Note that the transformations which are denoted as `Global` in the table, are applied only once to the set of all compatible attributes (altogether), whereas the transformations, which are denoted as `Local` are applied to: 1) every compatible attribute separately (one by one), and 2) all the set of compatible attributes (altogether). Indeed, transformations are not applied to combinations of attributes and hence there is no "combinatorial explosion". However, the user may apply the method several times in iteration, and as such, arrive to a combination that may induce better results. Yet this depends on the user and his/her availability to use the method iteratively.

The **impact** is the effect of transformations to the final result (i.e., predictive accuracy) of the selected algorithm, and it can be *predicted impact* or *computed impact*.

The *predicted impact* is calculated by applying the set of transformations, as defined above, and subsequently extracting the characteristics of the transformed datasets, to use them as inputs for predicting the performance of the respective algorithm on the transformed datasets.

The *computed impact* is calculated by similarly applying the set of transformations, but then subsequently applying the respective classification algorithm for real to the transformed datasets, and hence obtaining the real performance (e.g., predictive accuracy) of the classification algorithm on the transformed datasets. In terms of computational complexity, the latter is a costly process and it is performed only for the sake of evaluating the system.

The experiments were performed on an Intel Core i5 machine, running at 1.70 GHz with 8 GB of main memory. An experiment for a single algorithm, on average took approximately 4 CPU hours.

On each run, the system internally categorizes a transformation into one of the following three categories:

- *Good* — an improvement of the final result for the respective algorithm is predicted if the transformation were to be applied, compared to the prediction obtained on the non-transformed version of the dataset,
- *Bad* — a worsening of the final result for the respective algorithm is predicted if the transformation were to be applied, compared to the prediction obtained on the non-transformed version of the dataset,
- *Neutral* — neither improvement nor worsening is predicted if the transformation were to be applied.

The aim of the experiments is roughly to verify whether the categorizations made by the system are true for real (i.e., whether a transformation categorized as *Good*, is Good for real and improves the performance of the algorithm). This, as previously mentioned — though costly, is done by executing the data mining algorithms on the transformed datasets and computing the real impact of the transformations (cf. Figure 4.3).

In this context, we mark as *Successful* the cases (i.e., datasets) on which the *real/computed average improvement* we get from all the transformations categorized as *Good* for a dataset, is greater than the *real/computed average improvement* we get from the transformations that were categorized as *Bad* for the same dataset. That is, the transformations predicted as *Good* "beat" on average the transformations predicted as *Bad*. In contrast, we mark as *Unsuccessful* the cases on which the transformations predicted as *Good* cannot "beat" on average the transformations predicted as *Bad*.

4.5.2 Experimental Results

In Figure 4.4, we show the results obtained when *Random Forests* are used as meta-learners. In the figure we show the comparison between the number of *Successful* cases — the green bar, and the number of *Unsuccessful* cases — the red bar. In addition, the gray bar denotes the total number of cases (datasets) for which we performed the experiments on each respective algorithm. Furthermore, the bars highlighted with *dashes*, *dots*, and *back slashes* refer to the results for *predictive accuracy*, *precision*, and *AUC*, respectively. Notice that the results for *recall* are omitted due to the fact that identical values with *predictive accuracy* were obtained (see weighted recall in Weka[3]).

Observe that the sum of *Successful* (green) and *Unsuccessful* (red) cases does not coincide with the total number of datasets (gray). This is because for some datasets we either do not find *Good* transformations (14.3%), or we do not find *Bad* transformations (7.9%). This happens because the datasets already belong to the best or the worst leaves of all the internal trees of the *Random Forests*, hence there can be no transformations that can move them

[3]http://weka.sourceforge.net/doc.stable/weka/classifiers/Evaluation

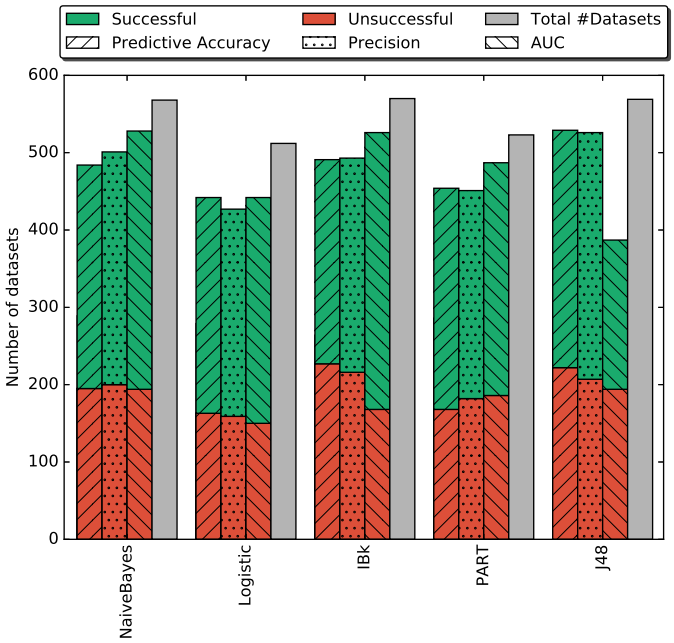


Fig. 4.4: Random Forest results

Table 4.5: Binomial significance test for Random Forests

	Predictive Accuracy			Precision			AUC		
Alg.	Su.	Su.+Un.	p-value	Su.	Su.+Un.	p-value	Su.	Su.+Un.	p-value
NB	289	484	7.40E-06	301	501	2.41E-06	334	528	3.36E-10
Log.	279	442	1.09E-08	268	427	4.34E-08	292	442	3.39E-12
IBk	264	491	4.31E-02	277	493	2.59E-03	358	526	0
PART	286	454	9.73E-09	269	451	1.61E-05	301	487	6.42E-08
J48	307	529	8.95E-05	319	526	3.79E-07	193	387	5.00E-01

to a better or worse leaf, respectively. As a matter of fact, in those particular cases we cannot compare the *Good* versus *Bad*, hence they do not appear neither as *Successful* nor as *Unsuccessful*.

In order to understand whether the numbers shown in the figure are significant, we performed a binomial distribution test comparing the number of *Successful* cases to the number of *Successful* + *Unsuccessful* cases with respect to the theoretical probability which is equal to `0.5`. The results obtained are shown in Table 4.5. The column *p-value* denotes how significant is the difference between the values of *Successful* and the population of *Successful* + *Unsuccessful*. We assume the difference to be significant if the *p-value* is less than or equal to `0.05`. Observe that our method gives significant values for all the algorithms and all the performance measures with the only exception of algorithm *J48* with performance measure *AUC*. Yet recall that when we performed feature selection using the partial correlation graphs between features and the response, we retained only the features that had a significant relationship within the limits of 0.05, which happens to be too restrictive for *J48* with *AUC* (i.e., some relevant feature is left out). If we increase the threshold to 0.1 (less restrictive) we obtain significant results for this case too. The *p-value* we obtain is 0.0011.

4.6 Related Work

A lot of research has been conducted in terms of providing user support for different steps of data analysis. The focus however, has usually been on the data mining step, and data pre-processing has generally been overlooked.

Weka [38], an open source tool for data mining, allows users to apply preprocessing algorithms but it does not provide assistance in terms of which

one to apply. However, since different data mining algorithms have different requirements regarding the dataset, some pre-processing is applied by default inside some of the algorithms. This pre-processing is usually a simple transformation that does not aim at improving the performance of an algorithm but it aims at transforming the dataset so that it can fit to the data mining algorithm. Furthermore, note that only few algorithm implementations in Weka contain these kind of on the fly transformations.

In AutoWeka [89], user assistance is provided, however only with regard to the data mining step. That is, the system suggests the best learning algorithm to use with it's proper parametrization without considering the pre-processing step. Hence, the user needs to deal with the pre-processing on his own.

In AmazonML[4], the system recommends an initial recipe for pre-processing, which is prepared taking into consideration the attributes of the dataset, including the response (i.e., the attribute to be predicted). The recipes provided are however pre-formatted instructions for common transformations and do not guarantee improvements of the final result. Hence, they are recommended only because they are applicable to the particular dataset, whereas we are interested in performing pre-processing with the only goal of improving the final result of the analysis.

eIDA [52], which is a product of the eLico[5] project, aims at autonomously constructing workflows that are combinations of pre-processing and data mining algorithms. In order to do that, the problem of workflow construction is viewed as a planning problem, in which a plan must be built consisting of operators that transform the initial data into models or predictions. In order to find the plans, an exhaustive combination of all applicable transformations with all applicable algorithms is performed. Taking into consideration the number of algorithms (e.g., hundreds in RapidMiner[6] — the project is built on top of RapidMiner), the search space of the problem is unfeasible to compute, hence the optimal solution may not be found. Moreover, in this approach, independent support exclusively for pre-processing is not provided. As a matter of fact, a *take it all or leave it* solution is given. In contrast, we focus only on pre-processing, which not only reduces the search space but at the same time allows independent support, where the data mining algorithm can be chosen at will.

[4]https://aws.amazon.com/machine-learning
[5]http://www.e-lico.eu
[6]http://rapidminer.com

There exist some other systems [15, 48, 63], however they also focus on providing support for the data mining step only.

4.7 Conclusions

In this chapter, we have shown that data pre-processing need not be treated as an independent, isolated step inside a knowledge discovery process. In contrast, it needs to be considered by the value it brings to the overall analysis. Hence, we attempted to do so by automatically suggesting pre-processing operators that aim at improving the overall analysis. This was made possible through meta-learning, which enables predicting the impact of transformations on the final performance of algorithms on the corresponding datasets, and in turn, allows ranking the transformations according to their impact on the final result.

We built a prototype solution that draws on a range of classification algorithms in Weka and makes it easy for non-experts to perform data pre-processing. An extensive evaluation on hundreds of datasets showed that for the set of algorithms considered, even blindly (e.g., users without any prior knowledge in data mining) applying the recommended transformations improves the final result of the algorithms on average. We believe that this can be a handy tool for experienced users as well, because they can discriminate within the recommended transformations and pick the ones that are potentially more suitable for their problem at hand.

5

Learning Based Recommending Assistant for Data Pre-processing

Data pre-processing is one of the most time consuming and relevant steps in a data analytics process (e.g., classification task). A given data pre-processing operator (e.g., transformation) can have positive, negative, or zero impact on the final result of the analysis. Expert users have the required knowledge to find the right pre-processing operators. However, when it comes to non-experts, they are overwhelmed by the amount of pre-processing operators and it is challenging for them to find operators that would positively impact their analysis (e.g., increase the predictive accuracy of a classifier). Existing solutions either assume that users have expert knowledge, or they recommend pre-processing operators that are only "syntactically" applicable to a dataset, without taking into account their impact on the final analysis. In this work, we aim at providing assistance to non-expert users by recommending data pre-processing operators that are ranked according to their impact on the final analysis. We developed a tool, PRESISTANT, that uses Random Forests to learn the impact of pre-processing operators on the performance (e.g., predictive accuracy) of 5 different classification algorithms, such as Decision Tree (J48), Naive Bayes, PART, Logistic Regression, and Nearest Neighbor (IBk). Extensive evaluations on the recommendations provided by our tool show that PRESISTANT can effectively help non-experts in order to achieve improved results in their analytic tasks.

5.1 Introduction

Although machine learning algorithms have been around since the 1950s, their initial impact has been insignificant. With the increase of data availability and computing power, machine learning tools and algorithms are making breakthroughs in very diverse areas. Their success has raised the need for mainstreaming the use of machine learning, that is, engaging even non-expert users (i.e., individuals with no proficiency in statistics and machine learning) to perform data analytics. However, the multiple steps involved in the data analytics process render this process challenging.

Data analytics as defined in [27], consists of data selection, data pre-processing, data mining, and evaluation or interpretation. A very important and time consuming step that marks itself out of the rest, is the data pre-processing step. Data pre-processing is challenging but at the same time has a heavy impact on the overall analysis. Specifically, it can have significant impact on the generalization performance of a classification algorithm [56, 63], where performance is measured in terms of the ratio of correctly classified instances (i.e., predictive accuracy).

The main tools used for data analysis (e.g., scikit-learn, R, Weka, AmazonML) overlook data pre-processing when it comes to assisting non-expert users in improving the overall performance of their analysis. These tools are usually meant for professional users who know exactly which pre-processing operators to apply. However, the staggeringly large number of available pre-processing operators (transformations) overwhelm non-expert users, and they require support.

Hence, our work focuses on assisting the users by reducing the number of pre-processing options to a bunch of potentially relevant ones. The goal is to retain only the transformations that have high positive impact on the analysis. Like this, we aim at reducing the time consumed in data pre-processing and at the same time improving the final results of the analysis. The focus is on classification problems, thus, our method recommends pre-processing operators that improve the performance of a given classification algorithm (e.g., increase the predictive accuracy).

Contributions. The main contributions of this chapter can be summarized as follows:

- We apply meta-learning techniques to develop a system that is capable of recommending pre-processing operators (transformations) that pos-

itively impact the final result of some classification tasks. Our method is based on training an algorithm to learn the impact of pre-processing operators and then use it to predict and ultimately rank different pre-processing operators.

- We perform an extensive experimental evaluation to compute the accuracy of the rankings with regards to: a) the whole set of transformations, and b) the top-*K*. For the former, we obtain an accuracy of 61% as an average for all the algorithms we consider. For the latter (i.e., K=1), the accuracy increases up to 68% on average.
- We evaluate our rankings with regards to the benefit/gain obtained from the user's point of view and we measure it using a classical information retrieval metric, Discounted Cumulative Gain (DCG). For the whole set of transformations we are as close as 73% to the gain obtained from the ideal rankings, whereas for the top-1 we are as close as 79%.
- We perform an empirical study comparing the ability of real users against our approach, on finding a transformation that positively impacts the classification accuracy on a set of randomly selected classification problems. Our approach, on average, performs 2.5 times better.

The remainder of this chapter is organized as follows. In Section 5.2, we give an overview on data pre-processing and we perform empirical analysis on the impact of pre-processing. In Section 5.3, we discuss meta-learning and its main components. In Section 5.4, we present our tool and proposed method on using meta-learning for data pre-processing. In Section 5.5, we provide an extensive evaluation of our approach. In Section 5.6, we discuss the related work and finally, in Section 5.7, we provide the conclusions.

5.2 Data Pre-processing

One of the reasons that makes data pre-processing consume a lot of time is that it encompasses a broad range of activities. Sometimes data needs to be transformed in order to fit the input requirements of the machine learning algorithm (e.g., if the algorithm accepts only data of numeric type, data is transformed accordingly) [49]. Sometimes data requires to be transformed from one representation to another (e.g., from an image representation to a

matrix representation) [32], or data may even require to be integrated with other data to be suitable for exploration and analysis [60]. Finally and more importantly, data may need to be transformed with the only goal of improving the performance of a machine learning algorithm [17]. The first two types of transformations are more of a necessity, whereas the latter is more of a choice, and since an abundant number of choices exist, it is time consuming to find the right one. In this chapter (and in the whole thesis), we target the latter type of pre-processing, and as such, the transformations taken into consideration are of the type that can impact the performance of data mining algorithms (i.e., classification algorithms). They are listed in Table 4.1 (cf. Chapter 4).

In Table 4.1, a transformation is described in terms of the *Technique* it uses, *Attributes* it takes, the *Input Type*, and also the *Output Type*. The transformations in Table 4.1 are the most commonly used ones in the Weka [38] platform, and their implementations are open source[1]. A short description for each category of transformations from Table 4.1 follows.

Discretization – the process of converting or partitioning continuous attributes to discretized or nominal/categorical attributes.

Nominal to Binary – the process of converting nominal/categorical attributes into binary numeric attributes.

Normalization – the process of normalizing numeric attributes such that their values fall in the range [0,1].

Standardization – the process of standardizing numeric attributes so that they have 0 mean and 1 variance.

Missing Value Imputation – the process of replacing missing values with some other value.

Principal Component Analysis – linear dimensionality reduction technique. The goal is to reduce the large number of directly observable features into a smaller set of indirectly observable features.

5.2.1 Overall Impact of Data Pre-processing

Other than theoretical analysis [56], to the best of our knowledge there is not much work on empirically studying the impact of pre-processing oper-

[1] https://github.com/bnjmn/weka

ators on real world classification problems. In order to assess the impact of pre-processing, we retrieved 533 datasets from the OpenML [92] repository and applied the pre-processing operators shown in Table 4.1. We used 5 different classification algorithms (i.e., J48, Naive Bayes, PART, Logistic, and IBk)[2] and measured their performance on the datasets before and after the transformations were applied. In Figure 5.1, we show the scatter plots of the relative change in predictive accuracy before and after the transformations where applied. In each scatter plot, we visualize the impact of all the transformations applied to datasets for a given algorithm, where green, red, and blue, denote positive, negative, and zero impact, respectively. The total set of transformations applied to all the datasets is approximately 25,000.

Transformations are applied depending on whether they are Local or Global (as classified in Table 4.1). If a transformation is Global it is applied only once to all the set of compatible attributes (e.g., normalizing all numeric attributes), whereas if it is Local, it is applied to: 1) every compatible attribute individually (e.g., discretizing one attribute at a time), and 2) all the set of compatible attributes (e.g., replacing missing values of all attributes).

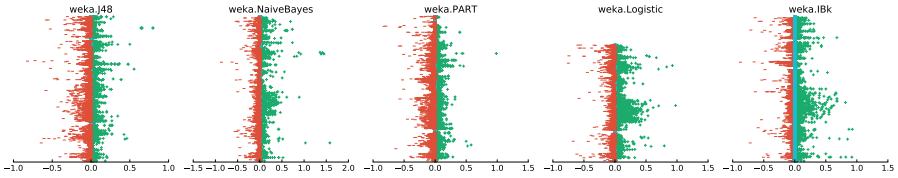


Fig. 5.1: Distributions of the (relative) impact induced by transformations in different algorithms

Observing Figure 5.1, one may conclude that:

- overall, transformations impact the final result of the analysis (i.e., they impact the predictive accuracy of the classification algorithms considered),
- the magnitude of the impact is heterogeneous, and
- there is no clear winner when it comes to the sign of the impact, i.e., transformations do not always impact positively or they do not always impact negatively.

[2]We chose one representative algorithm for 5 different classes of classification algorithms in Weka.

To confirm the latter, in Figure 5.2, we show the percentages of the positive, negative and neutral impacts using bar plots.

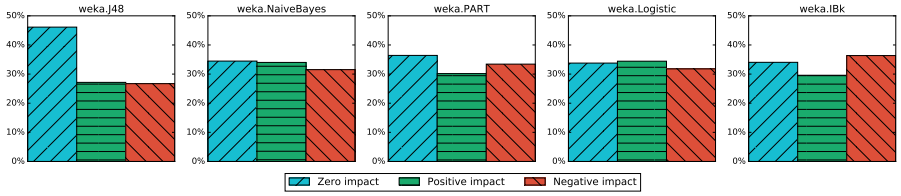


Fig. 5.2: The overall impact of transformations expressed in percentage for different algorithms

Figure 5.2, shows that transformations are almost uniformly distributed when it comes to the sign of impact, which intuitively leads to the conclusion that it may be challenging to distinguish among transformations that affect positively or negatively the final result.

5.2.2 Impact per Pre-processing Operator

In the previous section, we argued that in general, the impact of pre-processing is sound, but it may be difficult to find or predict the transformations that have positive impact. The analysis was performed on all transformations without distinguishing their types. The point now, is to check whether the previous conclusions still hold when we delve into studying categories of transformations separately (e.g., discretization), or conversely to the general picture, there exist some patterns (e.g., discretization has mainly positive impact).

In Figure 5.3, in a matrix like structure, we show the impact of every transformation from Table 4.1, for every algorithm considered. Circles are sized by the distance from a perfectly uniform distribution of the impact (i.e., 33% zero, 33% positive, 33% negative), and they are colored by the winner sign (i.e., blue if zero, green if positive, and red if negative impact is the winner). Thus, the bigger the circle and the sharper the color, the more obvious the pattern for a given transformation.

For instance, suppose Nearest Neighbor (IBk) for Normalization has a distribution of $80\%, 10\%, 10\%$ for zero, positive, and negative impact, respectively, and NaiveBayes for the same transformation has a distribution of $30\%, 45\%, 25\%$. Then, the sizes of the circles are determined by the euclidean distance between $(80, 10, 10)$ and $(33, 33, 33)$, for the first algorithm, and the distance between $(30, 45, 25)$ and $(33, 33, 33)$, for the second algo-

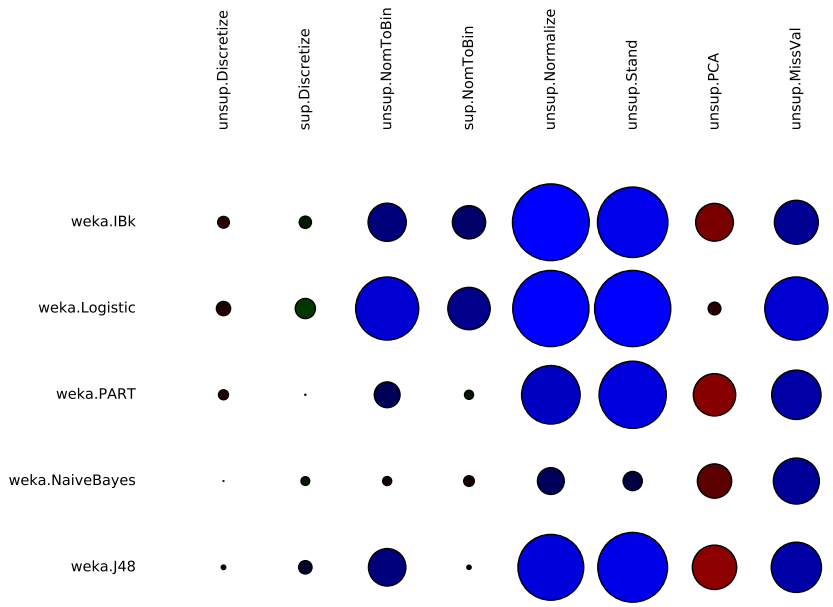


Fig. 5.3: Impact of pre-processing operators

rithm. The distance for the first algorithm (57.15) is obviously higher than the distance for the second algorithm (14.73), and hence the size of the circle. Furthermore, to define the colors of the circles, the distributions for negative, positive, and zero impact, participate proportionally to the RGB (red, green, blue) coloring scheme. Hence, for the above mentioned example for the first algorithm color blue will be more decisive, and for the second green. Yet, for the first algorithm, the color will be sharper than for the second, because of the values being higher.

The patterns emerging from the plot may help us in two directions. First, they can be used to devise basic rules or heuristics, i.e., if a transformation has a big blue circle for a given algorithm, then the transformation can be discarded because it is basically of no use for that particular algorithm, since most of the time it does not affect the final result. Secondly, they enable us to determine the more difficult transformations in terms of finding the impact to the final analysis, i.e., if a transformation has a small circle for a particular algorithm, it means that the distribution of the impact is close to uniform and hence a simple rule may not help in finding the impact of the transformation.

The latter rises the need for developing more sophisticated techniques for discovering the impact of transformations. To this end, we propose to learn the impact of transformations using meta-learning, and we delve into more details of this in Section 5.3.

Simple heuristics/rules as result of the empirical study

In Figure 5.3, circles of bigger size give clear patterns for devising simple heuristics. Notice that the bigger circles are usually of blue color. This means that the transformations for the corresponding algorithms are not of much use, since they do not impact the performance of the algorithms on the tested datasets. Some of the blue circles are obviously expected. For instance, it is well known that Normalization and Standardization do not impact the performance of Decision Trees (i.e., J48). Hence, a simple heuristic would be that, when a Decision Tree is chosen, Normalization and Standardization should not appear in the palette of possible transformations. The same holds for Logistic Regression. These transformations do not impact its performance. However, a counter-intuitive pattern appearing, is that of Normalization and Standardization with Nearest Neighbor. One would expect an impact of transformations, yet the circles are big and the color is blue, implying no impact.

We studied the internals of the Weka implementation in order to understand why was this happening. It resulted that Nearest Neighbor in Weka, internally uses a normalized Distance algorithm. Hence, the Normalization/Standardization is implicitly performed inside the learning algorithm and as a consequence, an external Normalization does not impact the performance. Similarly, for Logistic Regression we realized that NominalToBinary and Missing Value Imputation are applied implicitly in Weka.

The rest of the transformations have smaller circles in Figure 5.3, which implies that they have impact on the performance of algorithms. However, in the case of PCA for instance, although smaller, the circles are reddish. This indicates that although PCA impacts the performance, it needs to be carefully performed. A default or careless application of PCA may lead to a negative impact on the overall performance, which indicates that PCA should be used by experts. However, it is still interesting to discover the transformations that have negative impact and advise the non-expert user to avoid such transformations.

5.3 Meta-learning for Predicting the Impact of Data Pre-processing

Meta-learning is a general process used for predicting the performance of an algorithm on a given dataset. It is a method that aims at finding relationships between dataset characteristics and data mining algorithms [11]. Given the characteristics of a dataset, a predictive meta-model can be used to foresee the performance of a given data mining algorithm. For instance, in a classification problem, meta-learning can be used to predict the predictive accuracy of a classification algorithm on a given dataset and hence provide user support in the mining step.

Meta-learning can also be used to provide support in the pre-processing step [8, 35] (cf. Chapter 4). This can be done by learning the impact of data pre-processing operators (transformations) on the final result of the analysis. That is to say, detecting the transformations that have the most positive impact on the final analysis (i.e., transformations that after being applied on a dataset of classification type, increase the accuracy of a given classification algorithm on that dataset). This way, meta-learning pushes the user support to the data pre-processing step by enabling a ranking of transformations according to their relevance to the analysis. The ranking is made possible through the three phases, shown in Figure 3.1.

To enable meta-learning, the first thing to do for each classification algorithm is to create a dataset, that is, a matrix-like structure consisting of variables/features (predictors) and a response. The variables of this dataset are dataset characteristics — in this case characteristics of the transformed datasets (e.g., number of instances, number of missing values, etc.). The response is a performance metric of the classification algorithm (e.g., predictive accuracy). Since all these are metadata, this dataset is called a **meta-dataset**. Consequently its variables are referred to as **meta-features** and the response variable is named as **meta-response**. Furthermore, the process of learning on top of this meta-dataset is referred to as **meta-learning**, and the learning algorithm used is referred to as **meta-learner**. The meta-features, the meta-response, and the meta-learner are the key ingredients, and we will delve into details of each one of them, in the following sections.

5.3.1 Meta-features

The meta-features we specifically consider in this chapter are shown in Table 4.4. These are the set of meta-features extracted from OpenML [92].

In Table 4.4, column *Modifiable* indicates whether the meta-features are modifiable through the transformations we use (listed in Table 4.1). If meta-features are not modifiable/transformable, they are not considered because they remain constant and they do not reflect the impact of transformations.

Note that the ultimate goal is to predict the impact of transformations, and the impact per se, in this chapter is measured as the relative change of the performance of the algorithm before and after the transformation was applied. To this end, to the set of meta-features we consider, we attach also the base performance of the classification algorithm (i.e., the performance before the transformation is applied) and in addition we add features that capture the difference between the meta-features before and after the transformation was applied. We call these features **delta meta-features**. As a result, every meta-feature has its corresponding delta meta-feature. For instance, let us say that in a given dataset, before applying a transformation, the *number of continuous attributes* is 5. Assume we apply a transformation that is discretizing only one continuous attribute, then the number of continuous attributes becomes 4 and thus the delta of this feature is -1 (i.e., the *delta of the number of continuous attributes*).

Taking the deltas into account the total set of meta-features becomes large. We apply meta-feature extraction and selection in order to select only the most informative (with more predictive power) meta-features (cf Section 3.3.

5.3.2 Meta-response

The goal of meta-learning is to correctly predict the impact of transformations on the performance of machine learning algorithms. Different measures can be used to evaluate the performance of machine learning algorithms. Since we are dealing with classification problems, and hence the algorithms we consider are of classification type, the performance is usually measured in terms of *predictive accuracy*, *precision*, *recall*, or *AUC* [39]. Moreover, classification algorithms are usually evaluated using 10-fold cross-validation [53]. In Table 3.1, formulas for calculating these measures are given.

These measures are collected before and after the transformations are applied. The relative change between the performance obtained after the

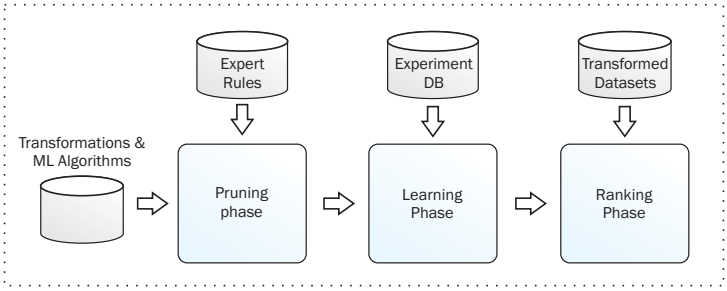


Fig. 5.4: Overview of PRESISTANT

transformation and the base performance (performance obtained before the transformation) is the impact of a transformation on the predictive power of a classification algorithm, and this is the meta-response. Based on the meta-features and delta meta-features mentioned previously, the goal of the meta-learner is to correctly predict this impact, which can be *positive* — if the transformation helps on improving the performance, *negative* — if the performance decreases after the transformation, and *zero* — if the performance remains the same.

5.3.3 Meta-learner

Given that meta-features (including delta meta-features) and meta-response candidates are defined, the next step is to define the meta-learning problem. Since the meta-response — the impact of transformations, is of continuous (numeric) type, the learning problem naturally fits to a regression type. Yet, we are interested in finding the transformations that either positively or negatively impact the data analysis, without necessarily needing to know the exact amount of impact. As a matter of fact, the problem may as well be defined as a classification problem, where three classes would be required, positive, negative, and zero.

5.4 PRESISTANT

When dealing with a classification problem, the non-expert data analyst has to choose from a large number of machine learning algorithms, and in addition, he/she encounters a plethora of different pre-processing options. Once the classification algorithm is chosen (i.e., one of the algorithms considered),

PRESISTANT assists the user by reducing the number of pre-processing options to only a set of relevant ones (i.e., operators that have positive impact). To do this, PRESISTANT uses a method consisting of three phases, shown in Figure 5.4 (cf. Appendix A for more details).

In the first phase, rules are applied to prune transformations such that the search space is reduced. In the second phase, a model is trained to learn the impact of transformations on the performance of classification algorithms. Finally in the third phase, the trained model is used to rank the newly arriving transformed versions of datasets.

5.4.1 Pruning Phase

Given that there is an overwhelming number of different transformations that can be applied to a dataset, in Section 5.2.2, we argued that simple rules can help on discarding transformations that have no impact. This translates to having a repository of Expert Rules (cf. Figure 5.4), that can be extended to contain any types of rules that may be known in advance and that reduce the number of potential transformations to be applied on datasets. Our first basic set of rules is derived from the experiments whose results were shown in Figure 5.3, where for instance we define rules in order to exclude Standardization and Normalization when considering algorithms like, IBk, Logistic, J48, and PART.

5.4.2 Learning Phase

Two important activities are performed in the learning phase. First, a meta-database (i.e., set of meta-datasets) is generated for all the classification algorithms considered (cf. Algorithm 1), and then on top of it, a learning algorithm is applied (cf. Algorithm 2). As a result, a statistical model (meta-model) is generated for every classification algorithm considered.

The inputs required to construct the meta-database are datasets, transformations — that are likely to improve the performance of classification algorithms, and the classification algorithms in consideration.

For the sake of simplicity, let us consider that we want to create the meta-dataset for a single classification algorithm. In line 5 of Algorithm 1, we first extract the dataset characteristics (i.e., meta-features from the original non-transformed datasets). Next, we apply all the available transformations to all the datasets and hence obtain transformed datasets, see line 8. We

Algorithm 1 Establish meta-database

Output: $meta\_db[1..\#algs][1..\#trans][1..\#metadata]$; ▷ meta-database; meta-dataset per classification alg.

```
1:  function CreateMetaDB(datasets[ ],transformations[ ],classAlgs[ ])
2:      metadata[ ] = ∅;                          ▷ the set of metadata to be collected
3:      for each algorithm alg in classificationAlgs do
4:          for each dataset ds in datasets do
5:              ds_mf[ ] = ComputeMetaFeatures(ds);                    ▷ Tbl. 4.4
6:              ds_pm = GetPerformanceWith10FoldCV(alg,ds);            ▷ Tbl. 3.1
7:              for each transformation tr in transformations do
8:                  trans_ds = ApplyTransformation(tr,ds);
9:                  trans_ds_mf[ ] = ComputeMetaFeatures(trans.ds);
10:                 Δmf[ ] = trans_ds_mf[ ] − ds_mf[ ];
11:                 trans_ds_pm = GetPerformanceWith10FoldCV(alg,ds);
12:                 mr = |trans_ds_mr − ds_pm| / ds_mr;                ▷ meta-response
13:                 metadata[ ] = trans_ds_mf[ ] ∪ Δmf[ ] ∪ ds_pm ∪ mr;
14:                 meta_db[alg][trans_ds] = metadata[ ];
15:             end for
16:         end for
17:     end for
18:     return meta_db;
19: end function
```

Algorithm 2 Create meta-models

Input: $datasets[..]$ ▷ available datasets of classification type
$transformations[..]$ ▷ set of transformations to be applied
$classificationAlgs[..]$ ▷ available classification algorithms
Output: $models[ ]$ ▷ meta-model for each algorithm

```
1:  function PerformMetaLearning
2:      meta_db = CreateMetaDB(datasets[ ],transformations[ ],classAlgs[ ]);
3:      meta_learner = RandomForest();                    ▷ a meta learner of choice
4:      models[ ] = ∅;
5:      for each algorithm alg in classificationAlgs do
6:          models[alg] = ApplyMetaLearner(meta_learner,meta_db[alg][ ][ ]);
7:      end for
8:      return models;
9:  end function
```

extract the meta-features from the transformed datasets in line 9, and take the difference between them and the meta-features from the original non-transformed datasets in line 10. Like this, we obtain the delta meta-features. Furthermore, to both original non-transformed datasets — line 6, and the

transformed ones — line 11, we apply the classification algorithm and then take the relative change between their corresponding performance measures (e.g., predictive accuracy) — line 12. The latter is the meta-response, which together with the meta-features of the non-transformed version of the dataset, the delta meta-features, and the performance measure of the original dataset compile the complete set of metadata — see line 13.

Once a meta-dataset for each classification algorithm is obtained, next a learning algorithm (i.e., meta-learner) is applied on top — line 6 of Algorithm 2, and as a result, a meta-model (i.e., statistical model) for each of the classification algorithms is obtained. PRESISTANT uses the Random Forest [12] algorithm as meta-learner. The XGBoost [16] algorithm was also tested as meta-learner. Similar results were obtained, yet Random Forest is easier to interpret.

5.4.3 Recommending Phase

The recommending phase starts when a user wants to analyze a dataset. He/she selects an algorithm to be used for the analysis and the system automatically recommends transformations to be applied, such that the final result is improved. This phase is described in Algorithm 3, where first the meta-features and the performance of the classification algorithm are extracted from the original non-transformed dataset in lines 4 and 5, respectively. Next, different transformations are applied to the dataset and from each transformed version of the dataset the necessary features (i.e., meta-features, delta meta-features) are computed — see lines 6-10. The extracted features are then fed to the predictor in line 11. The predictor in line 11, does no more than applying an already existing meta-model to the extracted features, to find the predicted impact of a transformation on the performance of the algorithm.

After the predicted impacts are obtained for all the transformations, they are ranked in descending order — using the probabilities of being positive; provided by the model, in line 12.

Note that a strong feature of PRESISTANT is that it actually only predicts the performance of the algorithms, otherwise the classification algorithms need to be applied for real on the transformed datasets. The former, as shown in Table 5.1, is orders of magnitudes faster than the latter.

Furthermore, the number of total transformations executed per dataset determines the level of interactivity that one can get with PRESISTANT. The

Algorithm 3 Recommend transformations

Input: $models[..]$; ▷ meta-model for each algorithm
$transformations[..]$; ▷ set of transformations to be applied
ds ▷ new dataset chosen by the user
Output: $transformations[..]$; ▷ trans. ordered according to predicted impact

```
1:  function RankTransformations(datasets[ ],transformations[ ],classAlgs[ ])
2:      predictions[ ][ ]; ▷ predictions for transformed datasets
3:      ds_mf[ ] = ComputeMetaFeatures(ds);
4:      ds_pm = GetPerformanceWith10FoldCV(alg,ds);
5:      for each transformation tr in transformations do
6:          trans_ds = ApplyTransformation (tr,ds);
7:          trans_ds_mf = ComputeMetaFeatures(ds);
8:          Δmf[ ] = trans_ds_mf[ ] − ds_mf[ ];
9:          features[ ] = trans_ds_mf[ ] ∪ Δmf[ ] ∪ ds_pm;
10:         predictions[tr] = ApplyModel(features[ ],models[classAlg]);
11:     end for
12:     transformations = RankByProbabilities(predictions,desc = true);
13:     return transformations;
14: end function
```

cost of executing most of the transformations individually, as shown in Table 5.2, is very cheap (notice that could be even cheaper executing several in a single pass over the dataset), unless more complex transformations like PCA are considered. However, as we already mentioned in Section 5.4.1, the task of an expert in the pruning phase would be to define rules that would filter the useless transformations and at the same time in a sense configure the overall level of interactivity.

Algorithm	Alg. Execution	Predictions
weka.J48	4658	0.1
weka.NaiveBayes	1530	0.1
weka.PART	14144	0.1
weka.Logistic	28880	0.1
weka.IBk	7624	0.1

Table 5.1: Comparison of average run times (ms) per dataset, real executions vs predictions

Transformation	Exec. Time
Discretization	14.65
NominalToBinary	4.63
Normalization	0.50
Standardization	0.63
PCA	980

Table 5.2: Average run time (ms) of transformations per dataset

5.5 Evaluation

We perform experimental studies to evaluate the performance of our tool. In particular, the aim of the experiments are three-fold:

1. Asses the performance of PRESISTANT in terms of the quality of the predictions from the meta-learner perspective. We try to answer the question "How good are the predictions?" (cf. Section 5.5.1).

2. Assess the gain obtained by the recommendations from the user perspective. We try to answer the question "How valuable are the recommendations?" (cf. Section 5.5.2).

3. Assess the performance of the recommendations of PRESISTANT, compared to the transformations picked by humans in a set of randomly selected classification problems. We try to answer the question "How difficult is to find the correct transformations in practice?" (cf. Section 5.5.3).

To enable the use of the entire set of datasets in the experiments, we use the 10-fold cross-validation method. This entails that for each classification algorithm considered, when building the meta-models, if a dataset is used in testing the same is not considered in the training.

Furthermore, we performed experiments treating the meta-learning problem both as classification and as regression. Similar results were obtained in both cases. We discuss only the results obtained with classification.

5.5.1 The Quality of Predictions

Predictions provided by the meta-model enable the ranking of transformations. The list of recommended transformations can be very large in case a lot of transformations are considered (e.g, with different parametrization). One may be interested in the whole set of transformations (e.g., "recommending all good items" in collaborative filtering), or only on the top-K transformations, K being an arbitrary number (e.g., "recommending some good items" in collaborative filtering). The latter is more realistic since the greater the ranked position, the less valuable a transformation is for the user, because the less likely it is that the user will examine the transformation due to time, effort, and cumulated information from transformations already seen/applied [43].

Table 5.3: Confusion matrices

Real \ Predicted	Positive	Negative	Zero
Positive	TP	FN_P	$F0_P$
Negative	FP_N	TN	$F0_N$
Zero	FP_0	FN_0	T0

a)

Real \ Predicted	Positive	Negative	Zero
Positive	TP	FNP	
Negative	FP′	TNP	
Zero			

b)

We performed evaluations both considering the whole set of transformations and considering only the top-K.

Evaluation of the quality of the whole set of transformations

When treated as classification, the meta-learning problem translates to a multi-class classification problem with three classes (i.e., positive, negative, zero) in the response variable. Given that it is a multi-class problem and knowing that all the classes do not have the same importance, we cannot use the traditional binomial confusion matrix for the evaluation. For instance, regarding the importance of classes, miss-predicting a zero transformation does not have the same impact as miss-predicting a positive or negative transformation.

Therefore, in Table 5.3a, we devise a confusion matrix that consists of two parts. The inner (green) part is the traditional confusion matrix for the positive and negative predictions and the outer (orange) part is the one that takes into account the zero class.

Furthermore, since datasets have varying numbers of attributes, they do not have the same number of transformations applied to them i.e., some datasets can have more transformations than others (cf. Table 4.1). To give equal importance to every dataset regardless of the number of transformations, we assign weights to their corresponding transformations (i.e., transformed versions of the same dataset): $w_{T_d} = 1/|T_d|$, where $|T_d|$ is the total number of transformations applied to dataset d. Like this, each dataset has weight equal to 1.

Using the matrix shown in Table 5.3a, per dataset d and for the set of its transformations T_d with their corresponding weights w_{T_d}, we evaluate our system calculating the Predictive accuracy (PA_d), Precision (Pr_d), Overall recall (OR_d), and G-measure (G_d), defined as follows:

$$PA_d = \frac{TP + TN}{TP + FN_P + FP_N + TN} \quad Pr_d = \left(\frac{TP}{TP + FP_N} + \frac{TN}{TN + FN_P} \right) /2$$

$$OR_d = \frac{TP + FN_P + FP_N + TN}{TP + FN_P + FP_N + TN + (F0_P + F0_N)} \quad G_d = 2 \left(\frac{PA_d \times OR_d}{PA_d + OR_d} \right)$$

where TP represents the number of true positives, FN_P the number of false negatives, FP_N the number of false positives, and TN the number of true negatives. Furthermore, $F0_P$ represents the number of transformations that are predicted as zero but in reality they are positive, $F0_N$ represents the zero predicted transformations that are in reality negative. Finally, FP_0 are positive predictions that in reality have zero impact, FN_0 negative predictions that in reality have zero impact, and $T0$ are the true zeros. Notice that FP_0 and FN_0 are less harmful than $F0_P$ and $F0_N$, respectively, since predicting a transformation as zero and then having a positive impact in real ($F0_P$) is worse than predicting a transformation as positive and then having zero impact in real (FP_0). The same applies for FN_0 when compared to $F0_N$.

The aforementioned measures are calculated for individual datasets d. Averaging the individual measures over all datasets with at least one relevant transformation, we obtain the mean Predictive accuracy PA, Precision Pr, Overall recall OR, and G-measure G. In the experiments performed on 533 datasets we obtained the results shown in Figure 5.5.

The results show that on average, if a user selects any transformation from the whole list of possible transformations (T_d), the system provides an accuracy of 63%.

Evaluation of the quality of the top-K recommendations

Since real users are usually concerned only with the top part of the recommendation list, a more practical approach is to consider the number of a datasets' relevant transformations ranked in the top-K positions.

Many details need to be considered in order to perform a proper evaluation of the K positions.

First of all, for the sake of simplicity, let us use the confusion matrix shown in Table 5.3b, where we denote as True Non Positive ($TNP = TN +$

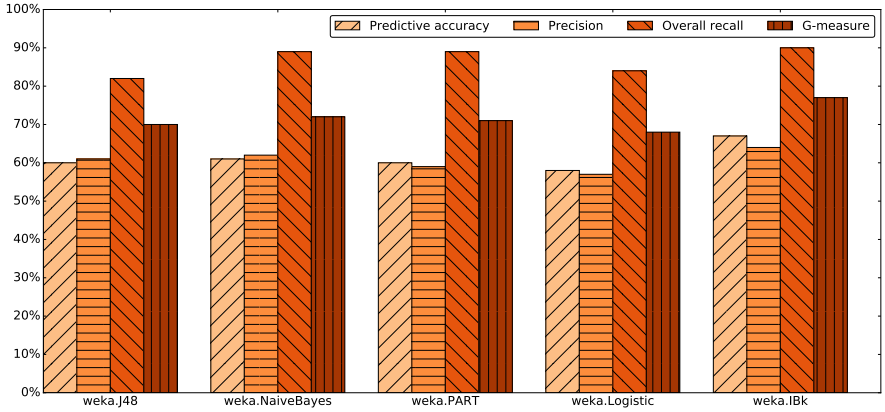


Fig. 5.5: Results obtained when evaluating the whole set of transformations

$F0_N + FN_0 + T0$), a transformation that is predicted as non-positive and it is non-positive in real (i.e, after executing the classification algorithm the transformation has no positive impact). False Non Positive ($FNP = FN_P + F0_P$), a transformation that is predicted as non-positive but is positive in real. Finally FP', a transformation that is predicted as positive but in reality can have either negative or zero impact.

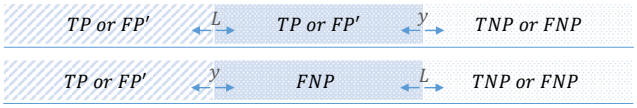
Next, notice that for each dataset there could be L transformations that have real positive impact, and the system (in practice) recommends y transformations that are predicted to have a positive impact.

To be able to perform evaluations for all the datasets, including the datasets with L=0 (i.e., datasets that do not have any transformations that have real positive impact), and to be able to calculate average measures for datasets with different L for all the positions in the ranking, we rank the transformations as follows: first we rank the y positively predicted transformations by their probability of being positive (the highest goes first). Next, we append the remaining real positive transformations (if any are left) up to L. Finally, we append all the remaining transformations ranked by their probability of being positive.

This ranking allows us to perform evaluations for each position K for any dataset with L real positive transformations. The results of the evaluations form a matrix of size $[L, K]$, and the possible evaluation we can have is shown in Table 5.4, where two possible scenarios are considered:

1. **if** $y > L$ (i.e., we predict too many positive transformations), the current

Table 5.4: Evaluation of our approach based on the chosen ranking (ordering of transformations)



transformation in position c can be below y where we can find either TP or FP', and above y where we can find either TNP or FNP.

2. **if** $y \leqslant L$ (i.e., we predict too few positive transformations), the current transformation in position c can be: below y, between y and L, and above L. Below y, we can find either TP or FP'. Above L, we can find either TNP or FNP. Finally, between y and L we can only find FNP, since here we have the transformations that are predicted as non-positive ($c > y$), but in reality they have positive impact.

Using the aforementioned, we can calculate accuracy measures where below the diagonal we can compute the ratio of true positives ($TP/(TP + FP' + TNP + FNP)$), and above the diagonal we can calculate the ratio of true non positives ($TNP/(TP + FP' + TNP + FNP)$).

In Table 5.5, we show the results obtained for algorithm IBk. The numbers shown inside each cell $[L, K]$, denoted as x/z, are the cumulative accuracies x, and the number of datasets z, which have at least K transformations and exactly L real positives. The cells are colored according to the values obtained, the darker the color the higher the accuracy.

Notice that the cells below the diagonal become darker as L grows. This means that we obviously perform better when the number of real positives is higher. Furthermore, the cells in the bottom part are darker than the rest. This means that we perform better in the top-K positions.

In Table 5.6, we show the weighted average results for all L, but only for position K=1[3].

Comparison with a random pick

To evaluate the performance of our approach, we compare it to the approach of a user randomly selecting a transformation to apply. For the latter, given the data, we need to find the probability of having TP below the diagonal and having TNP above the diagonal.

[3]The complete results for the corresponding algorithms are shown in Appendix A.

Table 5.5: Accuracy values obtained for the IBk classifier, for transformations in positions K in datasets with L real positive transformations. The numbers shown inside the cells, denoted as x/z, are the cumulative accuracies x, and the number of datasets z, which have at least K transformations and exactly L real positives. The last column, shows the values obtained after computing the weighted average for each position K for all possible L.

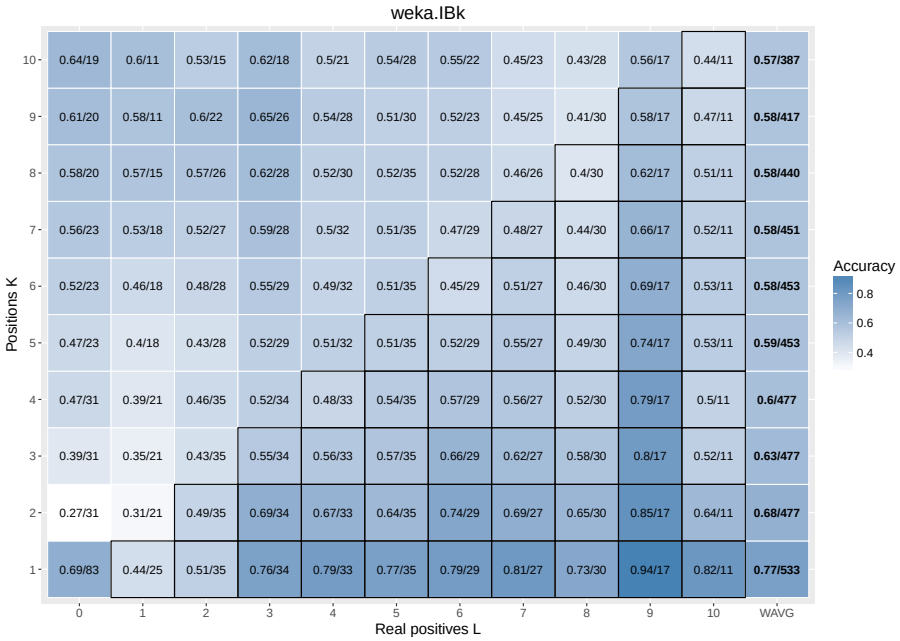
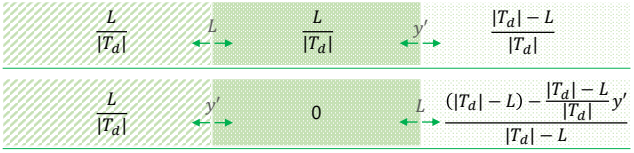


Table 5.6: Weighted average values for all L in K=1

Algorithm	WAVG in K=1	#Datasets
J48	0.62	533
Naive Bayes	0.78	533
PART	0.67	533
Logistic	0.57	480
IBk	0.77	533

Table 5.7: Evaluation of the random pick based on the chosen ranking



Finding the probability of having TP below the diagonal in the K^{th} position, translates to the problem of finding the probability of picking a positive transformation in K draws from a bag consisting of positive, negative, and zero (neutral) transformations. This follows a hyper-geometric distribution and the expected value of TP in a cell $[L, K]$ below the diagonal is calculated as $\mu_{TP} = K\frac{L}{|T_d|}$, where $|T_d|$ denotes the total number of transformations in dataset d. The expected value of TNP is calculated as $\mu_{TNP} = K\frac{L-|T_d|}{|T_d|}$.

To make a fair comparison with our approach, we need to assume the same ordering. Hence, the values to be calculated are shown in Table 5.7, where y' is the expected number of positive predictions a dataset can have, which is calculated as the ratio of the transformations of dataset d, given the proportion of all real positives we can have for algorithm a, $y' = |T_d|P_a(Positive)$ (probabilities for each algorithm can be found using the distributions of the impacts in Figure 5.2).

In Table 5.7, again two scenarios are considered:

1. **if** $y' > L$ (i.e., the random picks too many positive transformations), below y' we take the probability of being TP, and above y' we take the probability of being TNP.

2. **if** $y' \leqslant L$ (i.e, the random picks too few positive transformations), for the current transformation c positioned below y', we calculate the probability of being TP. For the transformation positioned between y' and L the probability of being TP is 0. Finally, for the transformation positioned above L we calculate the expected TNP based on the expected positive predictions y'.

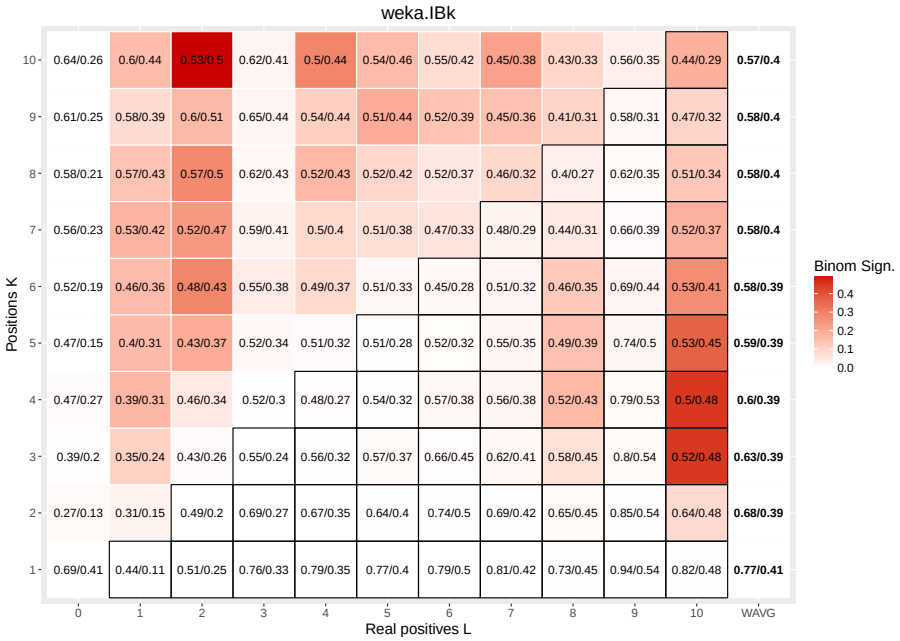
More precisely, given the conditions in Table 5.7, the probabilities for each cell $[L, K]$ are calculated using the following function:

$$P(L,K) = \begin{cases} (min(K,y')\frac{L}{|T_d|} + max(0,K-y')\frac{|T_d|-L}{|T_d|})/K, & \text{if } y' > L \\ (min(K,y')\frac{L}{|T_d|} + max(0,K-L)\frac{(|T_d|-L)-(\frac{|T_d|-L}{|T_d|}y')}{|T_d|-L})/K, & \text{if } y' \leqslant L \end{cases}$$

To show whether the values obtained by our approach are significant, we performed a binomial distribution test comparing the true positives obtained by our approach with the total number of datasets, against the theoretical probabilities obtained by the random pick. The results obtained are shown in Table 5.8, where the color of the cell denotes whether the value obtained is significant or not (white means significant). We consider the value to be significant if it is $p <= 0.001$.

It can be observed that significant values are obtained for most of the

Table 5.8: Significance values obtained when comparing results obtained with meta-learning and the random pick, for the IBk classifier. The numbers shown inside the cells, denoted as x/v, are the cumulative accuracies x obtained using our approach, and the random pick probabilities z, for datasets with at least K transformations and exactly L real positives. The last column shows the averages for our approach compared to the random pick, in each position K, weighted by the number of datasets in each L.



cells below the diagonal where the accuracy with regards to TP is measured. Furthermore, it is worth to mention that:

- Observing the bottom left-most cell, it can be noted that the system increases the chance of finding the transformations that do not positively impact the analysis (the system may suggest avoiding those transformations).
- The bottom right-most cell indicates that for position $K = 1$, we almost double the accuracy compared to the random pick (77% versus 41%). Moreover, the accuracy obtained for the whole set of transformations for IBk was 67%, and for top-1 becomes 77%.
- The probabilities of the random pick start to become higher above the diagonal due to the fact that it is easier to guess negative transformations as you go down in the ranking.
- Significance values are also impacted by the sample sizes (number of datasets), which are different for each L and they may also vary for different Ks. Yet, observe the last column where the weighted averages are shown. The calculations are done for all the datasets on each K, and the values obtained are significant.

5.5.2 The Gain Obtained from Recommendations

In the Information Retrieval domain, different measures that calculate the gain obtained from a ranked result have been proposed. The most popular one among them is the Discounted Cumulative Gain (DCG) [43]. The assumption is that the greater the ranked position, the less valuable the item (i.e., transformations) is for the user, because it is less likely that the user will examine it. Thus, DCG uses a discounting function that progressively reduces the gain as the rank increases. To compute DCG, a permutation/ordering π of the gain values G_{T_d} on the entire list of transformations in a given dataset d, results in the ordered list of gains G_{π,T_d} (a vector of length N, where N is $|T_d|$). In particular, we are interested in the permutation that sorts according to our predicted scores (i.e., predicted probabilities) for the various transformations in a dataset: G_{rec,T_d} denotes the recommended list or gains in descending order of the predicted scores of the transformations (highest ranked transformation is first in list). Moreover, the best (G_{best,T_d}) and worst (G_{worst,T_d}) possible permutations are also of interest.

For the general case DCG is computed as [87]:

$$DCG_{G_{\pi,T_d}} = \sum_{i=1}^{N} \frac{G_{\pi,T_d}[i]}{log_2(i+1)}$$

The calculations are performed for each dataset and we obtain values for the recommended ranking ($DCG_{G_{rec,T_d}}$), the best ranking ($DCG_{G_{best,T_d}}$) and the worst ranking ($DCG_{G_{worst,T_d}}$).

To obtain a relative value, we normalize the gain obtained by our recommendations in the following way:

$$nDCG_d = \frac{DCG_{G_{rec,T_d}} - DCG_{G_{worst,T_d}}}{DCG_{G_{best,T_d}} - DCG_{G_{worst,T_d}}}$$

This normalized value can be interpreted as a percentage of how close we are to the best ranking, and it is calculated for each dataset. Averaging the individual measures over all datasets with at least one relevant (non-neutral) transformation we obtain the mean $\overline{nDCG}$. This measure can again be calculated for the whole set of transformations or for the top-K. In Table 5.9, we provide the results obtained for the whole set of transformations and for top-1, for all the classification algorithms considered.

Table 5.9: Normalized discounted cumulative gain values

Algorithm	$\overline{nDCG}$		#Datasets[a]
	All trans.	Top-1	
J48	0.72	0.78	475
Naive Bayes	0.78	0.85	476
PART	0.73	0.79	476
Logistic	0.63	0.66	421
IBk	0.77	0.85	475

[a]Number of datasets with at least 1 relevant (non-neutral) transformation

5.5.3 PRESISTANT Compared to Humans

In this section, we discuss the results obtained in an empirical evaluation with humans. The importance of evaluations with real users has already been acknowledged in previous works [13].

Table 5.10: UCI datasets used in the experiments

Dataset	#Participants
autos	31
ecoli	37
diabetes	30
flags	39

Table 5.11: Average scores of users and PRESISTANT

Algorithm	Users	PRESISTANT
J48	0.10	0.27
Naive Bayes	0.52	1.0
PART	0.21	0.51
Logistic	0.10	0.50
IBk	0.17	0.55
Total	**0.22**	**0.57**

Our experiment, in the form of a quiz[4], is defined as follows: given a dataset, a classification algorithm, and a set of applicable transformations over the dataset, find the transformation that has the most positive impact on the classification accuracy of the algorithm. If none of the transformations is ought to have positive impact, pick option "None".

Among the possibly many positive transformations per dataset, users must find only one of them to have their answer considered as correct. On the other hand, PRESISTANT's choice is considered correct if its top recommended transformation is among the positive ones. In both cases, score 1 is assigned to the correct, and score 0 to the incorrect answer.

We performed experiments with 4 randomly selected datasets[5] (cf. Table 5.10), and the 5 classification algorithms that PRESISTANT supports. The maximum number of participants per dataset was 39, and their background varied between users with "No knowledge" in data mining — 18.71%, "Basic knowledge" — 48.75%, "Intermediate knowledge" — 31.07%, and "Expert knowledge" — 1.47%. Most of the participants held master's degrees in fields related to Computer Science — 72.83%. Others held Ph.D.'s in some Computer Science field — 15.89%, and the rest were undergraduate students in Computer Science — 11.28%. More than half of the participants were not students and they are currently pursuing their professional careers in either companies or academia, always in the field of data management and analytics.

The average scores obtained per algorithm by real users and PRESISTANT are compared in Table 5.11. PRESISTANT scored on average 2.5 times better than humans, showing its effectiveness in real scenarios. More interestingly

[4] http://www.essi.upc.edu/~bbilalli/presistant.html#quiz

[5] Datasets are retrieved from the UCI repository: https://archive.ics.uci.edu/ml/datasets.html

however, users performed worse than they were expected (i.e, they score 0.22 and the expected score with a random pick is close to 0.3). We suspect this occurred because of the fact that users were biased towards selecting transformations that are of type Global — applied globally to all the compatible attributes of a dataset. Indeed, 7 out of top 10 most frequently picked transformations were either of type Global or "None" — 57.08%. Yet, out of the transformations shown to the user on average only 37% of them were of type Global and the rest were Local.

Overall, the results indicate that in practice it is difficult to find the transformations that positively impact the analysis and that there is obvious need for user support.

5.6 Related Work

A lot of research has been done in order to address the problem of providing user support in the different steps of knowledge discovery. The idea has been to develop (semi) automatic systems that provide user assistance in one or many steps altogether. In the beginning, the focus has been to provide support exclusively for the data mining step. Recently however, the direction has shifted towards designing systems that specifically provide user assistance in the data pre-processing step and also systems that aim at fully automating the knowledge discovery process.

In the following, we give more details about the methods and systems developed to provide support in the different steps of knowledge discovery.

User support in data pre-processing. Since pre-processing covers a broad range of activities, different systems have been developed to tackle the problem of user assistance from different perspectives. There are systems that discover patterns and detect errors in data and then automatically infer relevant transformations. For instance, in Potter's Wheel [76], Wrangler [49], and Foofah [44], the relevant transformations are learned by example. The user either directly manipulates the visualized data or he/she needs to provide the output (to be) data.

In KATARA [18], DataXFormer [66], and VADA [54], they also infer transformations, however this time using external knowledge stored in knowledge bases, web tables, or knowledge obtained through interaction with crowds.

Other systems like NADEEF [21], Llunatic [34], and BigDansing [51], (semi) automate the detection and repairing of violations with respect to a

set of heterogeneous and ad-hoc constraints. Many types of quality constraints like functional dependencies, conditional functional dependencies, multivalue dependencies, and ETL rules can be defined. Their goal is to cope with multiple queries holistically and optimize their application.

In DataTamer [88] and DataCivilizer [23], they deal with the end to end curation (e.g., integration, de-duplication) of data from different sources.

Finally in ActiveClean [57], they aim at prioritizing the cleaning of records that are more likely to affect the results of the statistical modeling problems, assuming that the latter belong to the class of models called convex loss models (e.g., linear regression and SVMs).

Note that the assumption of the aforementioned systems is that an expert user is performing the analysis. That is, the user knows what the final shape of the transformed data should look like. Our assumption is that the user is a non-expert, and he/she wants to apply transformations only for the sake of improving the analysis. That is, he/she does not know what the input dataset to a mining algorithm should look like in order to yield better results/analysis.

User support for data mining (model selection). The main systems for providing support in data mining are dubbed as *Expert systems* and *Meta-learning systems.*

Expert systems [26, 75, 84] are the first and simplest systems to provide help to the user during the data mining phase. Their main component is a knowledge base consisting of expert rules. Given the input, either from the user or extracted from the dataset, rules are used to determine the mining algorithms to be recommended.

Meta-learning systems (MLS) [11, 36, 46] are more advanced than *Expert systems*. The rules that were statically defined by the experts in the previous category are dynamically learned here. MLSs try to discover the relationship between measurable features of the dataset and the performance of different algorithms, which is a standard learning problem. The learned model is then used to predict the most suitable data mining algorithm for a given dataset.

In general, the drawback of these systems is that they overlook the impact of pre-processing.

User support for data analytics (knowledge discovery). When it comes to automating the whole knowledge discovery process we distinguish between *Case-based reasoning systems*, *Planning-based data analysis systems*, and *AutoML (automated machine learning) systems*.

Case-based reasoning systems (CBS) [26, 61, 68] store the successfully applied workflows (i.e., machine learning pipelines) as cases, in a *case base*, with the goal of reusing them in the future. When faced with a new problem (i.e., dataset) provided by the user, these systems return k similar cases, which can be further adapted to the current problem.

Planning-based data analysis systems (PDA) [24, 52, 95] are able to autonomously design valid workflows without relying on similarities. To this end, the workflow composition problem is treated as a planning problem, where a plan is built by combining operators that transform the initial problem into accurate models or predictions. In order to construct valid workflows, the input, output, preconditions, and effects (IOPE) of each operator (e.g., pre-processing or data mining algorithm) need to be formally defined. Plenty of workflows are then generated by combining operators that syntactically complement one another.

AutoML systems [28, 71, 89] refer to systems that try to automatically optimize the hyperparameters of operators. The goal is to automatically generate workflows or machine learning pipelines that give optimal results for the task at hand. Typically, Bayesian optimization methods are used to tune and optimize the hyperparameters. Since Bayesian optimization is randomized and it starts from a random configuration of hyperparameters, meta-learning has been used to find a good seed for the search [29].

Note that the full automation of knowledge discovery has been an ultimate goal of many research works. However, reaching such an automation has shown to be computationally expensive due to the search space, which explodes even more with the increase of pre-processing and mining operators. Therefore, the usability of such approaches in realistic scenarios is limited. In addition, our approach of finding a bunch of relevant transformations can be seen as complementary to these solutions, since it can help in pruning their large search space.

5.7 Conclusions

In this chapter, we addressed the problem of assisting non-expert users to perform pre-processing with the goal of improving the final results of their classification tasks.

To provide assistance, we trained a model that learned the relationship between pre-processing operators and the performance of classification algo-

rithms. To this end, we were able to rank the transformations according to their impact on the final result of the analysis (i.e., the impact of transformations on the predictive accuracy of a classification algorithm). An extensive evaluation on hundreds of datasets and a set of classification algorithms showed that our approach gives promising results. More specifically, we were able to observe that:

- even if a user randomly picks a transformation from the entire list of transformations ranked by PRESISTANT we obtain an average accuracy of 61%, for all the algorithms considered,
- recommending only the top-1 transformation, increased the average accuracy to 68%,
- measuring the gain obtained from our ranking for all transformations using DCG, we were as close as 73% on average to the gain obtained from the best possible ranking (for all the algorithms considered),
- measuring the gain from the top-1 recommendations using DCG, we were as close as 79% on average to the gain obtained from the best possible ranking,
- in a set of randomly selected classification problems, PRESISTANT performed 2.5 times better than humans (mostly non-experts).

Finally, the results indicate that PRESISTANT can assist users to more effectively identify the pre-processing operators appropriate to their applications, and to achieve improved results.

6

Conclusions and Future Directions

6.1 Conclusions

When data pre-processing is treated as an independent, stand-alone process, its impact on a potential analysis (e.g., mining) is not questioned, and this is normal. The problem rises when data pre-processing is performed within a knowledge discovery process (i.e., it is performed as a step before mining), and yet its impact on the analysis is not considered. Systems aiming at providing user support in data pre-processing have typically considered it as an independent step, and therefore their support has generally been syntactic.

Differently from previous works, the goal of this thesis was to provide user assistance in data pre-processing with the aim of improving the performance of the overall analysis. To narrow down the problem, we focused on classification problems. That is, given the classification algorithm to be used, the task was to provide user support by recommending data pre-processing operators that would improve the results of the analysis (e.g., increase the predictive accuracy).

To this end, we proposed the use of *meta-learning* as a method that exploits metadata about data mining experiments on transformed datasets, to learn the relationships between pre-processing operators and classification algorithms. Since this was a learning problem that naturally translates to a

knowledge discovery process, we had to pass through all the steps of knowledge discovery (data analytics) in order to make it work. Therefore, as mentioned in Chapter 1 (cf. Figure 1.3), the overall work of this thesis matches a knowledge discovery process, and hence the different chapters map to its different steps.

In the following, we provide conclusions for each step (i.e., each chapter), separately.

Metadata selection. In Chapter 2, we were generally concerned with the metadata used within different methods that aimed at providing user assistance either for the whole knowledge discovery process, or for each of its steps separately. To this end, we analyzed different systems that are referred to as *Intelligent Discovery Assistants* and identified the specific metadata they use for enabling user support. We classified the metadata found, and identified additional metadata that was not exploited by these systems, and argued for their potential benefits. Furthermore, we developed a metadata repository for storing such metadata and proposed an initial architecture that could make use of such metadata. Since this chapter dealt with all the possible metadata that can be selected, it fitted to the first step of (meta)data selection in a knowledge discovery step.

Metadata pre-processing. In Chapter 3, we focused specifically on the metadata used in meta-learning and we performed *exploratory analysis* with the goal of selecting the most predictive subset of metadata. Particularly, in this chapter, we used a method that consists of two phases, namely meta-feature *extraction* and *selection*. In the first phase, we applied Principal Component Analysis in order to discover the latent concepts behind the meta-features. This allowed to study the metadata in a higher level, and at the same time enabled to have more robust features. Furthermore the latent features are in a sense more independent from the actual data, because they can generalize to future datasets by omitting not existing meta-features or including new ones, provided that they are related with the actual concepts present in the latent features. Next, in the second phase we used partial correlations graphs to select the most predictive latent-features, which in our evaluations resulted to improve the performance of the meta-learning. In addition, in this chapter we provided a novel way of visualizing the relationship between different meta-features and latent-features. Since this chapter

deals with the pre-processing of metadata, it automatically maps to the second step of knowledge discovery.

Meta-learning. In Chapter 4, which maps to the mining phase of the knowledge discovery process, we proposed to use meta-learning with the aim of providing user support in the data pre-processing step. Therefore, the metadata that was pre-processed in the previous step was used in this step with the aim of building meta-models for each classification algorithm considered. Specifically, the goal was to learn the relationship between transformations and the performance (e.g., predictive accuracy) of the classification algorithms. Therefore, using the characteristics of the transformed datasets as meta-features, we were able to predict the performance of a classification algorithm on the transformed dataset. Since the goal was to predict a continuous value, our prediction (meta-learning) problem was of regression type. The predictions eventually allowed to rank transformations according to their relevance to the result of the analysis. Our evaluation on hundreds of datasets from OpenML, showed that this approach gives significant results. More precisely, the binomial comparison between the *successful* and *unsuccessful* cases for different classification algorithms and different performance measures gave significant values (cf. Chapter 4). We considered the values to be significant if the *p-value* obtained was below 0.05.

Interpretation *and looping back to pre-processing.* The last phase of a knowledge discovery process is evaluating and interpreting the results obtained from the analysis (mining). Similarly in this thesis, in Chapter 5, we reached the point of interpretation, where we realized that there was space for improvement with respect to the results obtained from the mining (meta-learning) step. Therefore as a result, we "looped back to the pre-processing step". That is, we once more studied the meta-features used for meta-learning and we decided to extend them. Particularly, we added the performance of the classification algorithm on the non-transformed version of the dataset as a new meta-feature and we also considered new features like the ones obtained when taking the difference between the characteristics of the transformed dataset and the base dataset. We named these features as *delta meta-features*. Furthermore, differently from the previous chapter, this time our task was to predict the *impact* of transformations, which was measured as the

relative change that a transformation induces on the performance of a classification algorithm (e.g., predictive accuracy). This value could be either *positive*, *negative*, or *zero*. Thus, the meta-learning problem translated to a classification problem. Using this approach, we built a tool, PRESISTANT, that given a classification algorithm recommends transformations that positively impact the analysis. We extensively evaluated our recommendations from three perspectives. In the first one, we checked how accurate our predictions were. In the second, we analyzed how much gain they provided to the final non-experienced user. Finally in the third, we analyzed the performance of PRESISTANT compared to humans in a realistic scenario. The results showed that our approach was feasible, especially because our top first recommendations provided both good accuracy and gain when compared to an ideal ranking. Briefly, the evaluation of the rankings with regards to the gain obtained from the user's point of view, using a classical information retrieval metric — Discounted Cumulative Gain [43], showed that for the whole set of possible transformations of a dataset, PRESISTANT was as close as 73% to the gain obtained from the ideal rankings. Whereas for the best transformation in the ranking, PRESISTANT was as close as 79% on average for all the considered algorithms.

Finally, in summary, we contend that this thesis is a step towards bringing data pre-processing closer to non-experts, since even without any knowledge for data pre-processing this work makes it possible to find transformations that would positively impact the analysis.

6.2 Future Directions

Knowledge discovery is an inherently complex process that requires user interaction and intervention. Therefore, claiming full automation is rather optimistic, not to say impossible. This is mainly due to the important role that domain knowledge plays in knowledge discovery. Its need, effectiveness, and importance in all the stages of knowledge discovery has already been confirmed in the past research efforts. However, there is lack of methods that enable a comprehensive exploitation and definition of domain knowledge, for providing user support in a knowledge discovery process. A partial solution to this in this thesis was the module that allows to capture/introduce domain knowledge specifically for data pre-processing, through the defini-

tion of expert rules. However, we contend that this is not enough and there need to be more comprehensive and generic ways to formally define and use domain knowledge for user support in knowledge discovery. One option may be the use of semantic web technologies for developing frameworks that can systematically incorporate domain knowledge in an intelligent discovery environment [50, 58].

When it comes to the specific task of user support in data pre-processing, we currently consider the recommendation of single pre-processing operators and thus a combination of different operators can be achieved only through an iterative application of our approach. Yet we contend that this may be a limitation and thus may be tackled through user support in the form of workflows (i.e., connected operators). However, the latter implies the need for more meta-knowledge. That is, there is need for new ways of characterizing workflows such that they can be stored and used for learning on top of them [70].

Another aspect for future research is the characterization of datasets and the metadata used for meta-learning in data pre-processing. In Chapter 5, we showed that considering additional different meta-features, we can improve the performance of meta-learning for pre-processing. This was an indication that there may be room for even more improvement. To this end, there is need for engineering new meta-features that can better capture the relationship between transformations, algorithms, and datasets. This may include developing additional methods for characterizing data mining algorithms (e.g., through their prerequisites/preconditions) and data pre-processing operators.

Furthermore, another aspect that can be considered as a future direction is with regards to decreasing the computational cost incurred when applying different transformations to a dataset. More precisely, in our approach, we first need to execute all the transformations for which we aim to predict the impact. Since the number of transformations may be large, this can incur some computational cost. Hence, in this regard there is need for methods that ideally, foresee the impact or the change that transformations imply on datasets, without the need of physically executing them.

Finally, our study was limited to classification problems. That is, we are providing user support in data pre-processing only if the problem at hand is of classification type. However, our method can be directly extended to regression problems since they also belong to the domain of Predictive Ana-

lytics. In that case however, instead of considering the improvement of accuracy due to transformations, one needs to consider the improvement in the R^2 (in the cross-validation). Furthermore, for other data mining techniques like clustering and association rules, which belong to the domain of Descriptive Analytics, although not very obvious, one can define metrics that measure the quality of the obtained results and then extend our method. Possible metrics to be used can be *intervariability of centroids* for clustering, and *lift* for association rules.

Bibliography

[1] K. Baba, R. Shibata, and M. Sibuya. "Partial Correlation and Conditional Correlation as Measures of Conditional Independence". In: *Australian and New Zealand Journal of Statistics* 46.4 (2004), pp. 657–664 (cit. on p. 70).

[2] H. Bensusan and C. Giraud-Carrier. "Casa Batló is in Passeig de Grácia or how landmark performances can describe tasks". In: *Proceedings of the ECML Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*. 2000, pp. 29–46 (cit. on p. 39).

[3] H. Bensusan and C. G. Giraud-Carrier. "Discovering Task Neighbourhoods Through Landmark Learning Performances". In: *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. 2000, pp. 325–330 (cit. on p. 59).

[4] H. Bensusan, C. G. Giraud-Carrier, and C. J. Kennedy. "A Higher-order Approach to Meta-learning". In: *Proceedings of the International Conference on Inductive Logic Programming*. 2000 (cit. on p. 59).

[5] H. Bensusan and A. Kalousis. "Estimating the Predictive Accuracy of a Classifier". In: *Proceedings of the European Conference on Machine Learning*. 2001, pp. 25–36 (cit. on p. 59).

[6] A. Bernstein, F. J. Provost, and S. Hill. "Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification". In: *IEEE Transactions on Knowledge and Data Engineering* 17.4 (2005), pp. 503–518 (cit. on pp. 11, 18).

[7] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. "Automated Data Pre-processing via Meta-learning". In: *Proceedings of the International Conference on Model and Data Engineering*. 2016, pp. 194–208 (cit. on p. 72).

[8] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. "Intelligent assistance for data pre-processing". In: *Computer Standards & Interfaces* 57 (2018), pp. 101–109 (cit. on pp. 34, 91).

[9] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. "Towards Intelligent Data Analysis: The Metadata Challenge". In: *Proceedings of the International Conference on Internet of Things and Big Data*. 2016, pp. 331–338 (cit. on p. 35).

[10] P. Brazdil, J. Gama, and B. Henery. "Characterizing the Applicability of Classification Algorithms Using Meta-level Learning". In: *Proceedings of the European Conference on Machine Learning*. 1994, pp. 83–102 (cit. on pp. 7, 59).

[11] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. 1st ed. Springer Publishing Company, Incorporated, 2008 (cit. on pp. 7, 11, 34, 65, 91, 110).

[12] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32 (cit. on p. 96).

[13] J. Carver, M. L. Jaccheri, S. Morasca, and F. Shull. "Using Empirical Studies during Software Courses". In: *Empirical Methods and Studies in Software Engineering, Experiences from ESERNET*. Vol. 2765. Springer, 2003, pp. 81–103 (cit. on p. 107).

[14] C. Castiello, G. Castellano, and A. M. Fanelli. "Meta-data: Characterization of Input Features for Meta-learning". In: *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence*. 2005, pp. 457–468 (cit. on pp. 35, 59).

[15] M. Charest et al. "Bridging the Gap Between Data Mining and Decision Support: A Case-based Reasoning and Ontology Approach". In: *Intelligent Data Analysis* 12.2 (Apr. 2008), pp. 211–236 (cit. on p. 81).

[16] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA, 2016, pp. 785–794 (cit. on p. 96).

[17] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. "Data Cleaning: Overview and Emerging Challenges". In: *Proceedings of the International Conference on Management of Data*. 2016, pp. 2201–2206 (cit. on pp. 86, 133).

[18] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. "KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing". In: *Proceedings of the International Conference on Management of Data*. 2015, pp. 1247–1261 (cit. on p. 109).

[19] S. F. Crone, S. Lessmann, and R. Stahlbock. "The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing". In: *European Journal of Operational Research* 173.3 (2006), pp. 781–800 (cit. on pp. 62–64).

[20] CWM. *Object Management Group: Common Warehouse Metamodel Specification*. Available at http://www.omg.org/spec/CWM/1.1/PDF/. 2003 (cit. on p. 30).

[21] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. "NADEEF: A Commodity Data Cleaning System". In: *Proceedings of the International Conference on Management of Data*. 2013, pp. 541–552 (cit. on p. 109).

[22] T. Dasu and T. Johnson. *Exploratory data mining and data cleaning*. Vol. 479. John Wiley & Sons, 2003 (cit. on p. 64).

[23] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. "The Data Civilizer System". In: *Online Proceedings of the Biennial Conference on Innovative Data Systems Research*. 2017 (cit. on p. 110).

[24] C. Diamantini, D. Potena, and E. Storti. "Ontology-Driven KDD Process Composition". In: *Proceedings of the International Symposium of Advances in Intelligent Data Analysis*. 2009, pp. 285–296 (cit. on pp. 25, 111).

[25] J. B. Elsner and T. H. Jagger. "Prediction Models for Annual U.S. Hurricane Counts". In: *Journal of Climate* 19 (2006), p. 2935 (cit. on p. 1).

[26] R. Engels. "Planning tasks for KDD; Performing Task-Oriented User-Guidance." In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 170–175 (cit. on pp. 25, 110, 111).

[27] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "From Data Mining to Knowledge Discovery in Databases". In: *AI Magazine* 17.3 (1996), pp. 1–34 (cit. on pp. 1, 2, 84).

[28] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. "Efficient and Robust Automated Machine Learning". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2962–2970 (cit. on pp. 3, 7, 111).

[29] M. Feurer, J. T. Springenberg, and F. Hutter. "Initializing Bayesian Hyperparameter Optimization via Meta-Learning". In: *Proceedings of the Conference on Artificial Intelligence*. 2015, pp. 1128–1135 (cit. on pp. 7, 111).

[30] N. Foshay et al. "Does Data Warehouse End-user Metadata Add Value?" In: *Communications of the ACM* 50.11 (2007), pp. 70–77 (cit. on pp. 25, 26, 29).

[31] M. Friendly. *Milestones in the history of thematic cartography, statistical graphics, and data visualization*. 2009 (cit. on p. 1).

[32] T. Furche, G. Gottlob, L. Libkin, G. Orsi, and N. W. Paton. "Data Wrangling for Big Data: Challenges and Opportunities". In: *Proceedings of the International Conference on Extending Database Technology*. 2016, pp. 473–478 (cit. on pp. 86, 133).

[33] J. Fürnkranz and J. Petrak. "An Evaluation of Landmarking Variants". In: *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*. 2001, pp. 57–68 (cit. on p. 59).

[34] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. "The LLUNATIC Data-cleaning Framework". In: *Proceedings of the VLDB Endowment* 6.9 (July 2013), pp. 625–636 (cit. on p. 109).

[35] I. Gemp, G. Theocharous, and M. Ghavamzadeh. "Automated Data Cleansing through Meta-Learning". In: *Proceedings of the International Conference on Artificial Intelligence*. 2017, pp. 4760–4761 (cit. on p. 91).

[36] C. Giraud-Carrier. "The data mining advisor: meta-learning at the service of practitioners". In: *Proceedings of the International Conference on Machine Learning and Applications*. 2005, pp. 113–119 (cit. on pp. 25, 39, 110).

[37] A. Guazzelli, M. Zeller, W.-C. Lin, and G. Williams. "PMML: An Open Standard for Sharing Models". In: *The R Journal* 1 (2009), pp. 60–65 (cit. on p. 74).

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et al. "The WEKA Data Mining Software: An Update". In: *ACM SIGKDD Explorations Newsletter* 11.1 (Nov. 2009), pp. 10–18 (cit. on pp. 79, 86, 132).

[39] D. J. Hand. "Measuring classifier performance: a coherent alternative to the area under the ROC curve". In: *Machine Learning* 77.1 (2009), pp. 103–123 (cit. on p. 92).

[40] 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*. Vol. 358. Studies in Computational Intelligence. Springer, 2011, pp. 273–315 (cit. on p. 7).

[41] T. K. Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3 (2002), pp. 289–300 (cit. on p. 22).

[42] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441 (cit. on pp. 41, 69).

[43] K. Järvelin and J. Kekäläinen. "IR Evaluation Methods for Retrieving Highly Relevant Documents". In: *Proceedings of the International Conference on Research and Development in Information Retrieval*. 2000, pp. 41–48 (cit. on pp. 98, 106, 116, 134).

[44] Z. Jin, M. R. Anderson, M. Cafarella, and H. V. Jagadish. "Foofah: A Programming-By-Example System for Synthesizing Data Transformation Programs". In: *Proceedings of the International Conference on Management of Data*. 2017, pp. 1607–1610 (cit. on pp. 2, 109).

[45] H. F. Kaiser. "The varimax criterion for analytic rotation in factor analysis". In: *Psychometrika* 23.3 (1958), pp. 187–200 (cit. on pp. 43, 70).

[46] A. Kalousis. "Algorithm Selection via Meta-learning. University of Geneve". In: Ph.D. Dissertation. 2002 (cit. on pp. 7, 110, 133).

[47] A. Kalousis and M. Hilario. "Feature Selection for Meta-learning". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. 2001, pp. 222–233 (cit. on pp. 35, 59).

[48] A. Kalousis and M. Hilario. "Model Selection via Meta-Learning: A Comparative Study". In: *International Journal on Artificial Intelligence Tools* 10.4 (2001), pp. 525–554 (cit. on pp. 25, 66, 81).

[49] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. "Wrangler: Interactive Visual Specification of Data Transformation Scripts". In: *Proceedings of the International Conference on Human Factors in Computing Systems*. 2011, pp. 3363–3372 (cit. on pp. 2, 85, 109, 133).

[50] C. M. Keet, A. Lawrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, and M. Hilario. "The Data Mining OPtimization Ontology". In: *Journal of Web Semantics* 32 (2015), pp. 43–53 (cit. on p. 117).

[51] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. "BigDansing: A System for Big Data Cleansing". In: *Proceedings of the International Conference on Management of Data*. 2015, pp. 1215–1230 (cit. on p. 109).

[52] J. Kietz, F. Serban, S. Fischer, and A. Bernstein. "Semantics Inside! But Let's Not Tell the Data Miners: Intelligent Support for Data Mining". In: *Proceedings of the Extended Semantic Web Conference*. 2014, pp. 706–720 (cit. on pp. 25, 80, 111).

[53] R. Kohavi. "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 1995, pp. 1137–1143 (cit. on pp. 39, 71, 72, 92).

[54] N. Konstantinou, M. Koehler, E. Abel, C. Civili, B. Neumayr, E. Sallinger, A. A. Fernandes, G. Gottlob, J. A. Keane, L. Libkin, and N. W. Paton. "The VADA Architecture for Cost-Effective Data Wrangling". In: *Proceedings of the International Conference on Management of Data*. 2017, pp. 1599–1602 (cit. on p. 109).

[55] I. Kopanas, N. M. Avouris, and S. Daskalaki. "The Role of Domain Knowledge in a Large Scale Data Mining Project". In: *Proceedings of the Hellenic Conference on Methods and Applications of Artificial Intelligence*. 2002, pp. 288–299 (cit. on p. 23).

[56] S. B. Kotsiantis and et al. *Data Preprocessing for Supervised Learning*. 2006 (cit. on pp. 84, 86).

[57] S. Krishnan, M. J. Franklin, K. Goldberg, J. Wang, and E. Wu. "ActiveClean: An Interactive Data Cleaning Framework For Modern Machine Learning". In: *Proceedings of the International Conference on Management of Data*. 2016, pp. 2117–2120 (cit. on p. 110).

[58] A. Lawrynowicz. *Semantic Data Mining - An Ontology-Based Approach*. Vol. 29. Studies on the Semantic Web. IOS Press, 2017 (cit. on p. 117).

[59] C. Lemke, M. Budka, and B. Gabrys. "Metalearning: a survey of trends and technologies". In: *Artificial Intelligence Review* 44.1 (2015), pp. 117–130 (cit. on pp. 7, 34).

[60] M. Lenzerini. "Data Integration: A Theoretical Perspective". In: *Proceedings of the International Symposium on Principles of Database Systems*. 2002, pp. 233–246 (cit. on pp. 86, 133).

[61] G. Lindner and R. Studer. "AST: Support for Algorithm Selection with a CBR Approach". In: *European Conference on Principles and Practice of Knowledge Discovery in Databases*. 1999, pp. 418–423 (cit. on pp. 25, 111).

[62] S. Martin, S. Vora, K. Yuen, and M. M. Trivedi. "Dynamics of Driver's Gaze: Explorations in Behavior Modeling Maneuver Prediction". In: *IEEE Transactions on Intelligent Vehicles* PP.99 (2018), pp. 1–1 (cit. on p. 1).

[63] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, eds. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994 (cit. on pp. 58, 81, 84).

[64] R. Miotto, L. Li, and J. T. Dudley. "Deep Learning to Predict Patient Future Diseases from the Electronic Health Records". In: *Proceedings of the European Conference on Information Retrieval*. 2016, pp. 768–774 (cit. on p. 1).

[65] M. Morchid, R. Dufour, P. Bousquet, G. Linarès, and J. Torres-Moreno. "Feature selection using Principal Component Analysis for massive retweet detection". In: *Pattern Recognition Letters* 49 (2014), pp. 33–39 (cit. on pp. 40, 41).

[66] J. Morcos, Z. Abedjan, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. "DataXFormer: An Interactive Data Transformation Tool". In: *Proceedings of the International Conference on Management of Data*. 2015, pp. 883–888 (cit. on pp. 2, 109).

[67] L. Moreau et al. "The Open Provenance Model core specification (v1.1)". In: *Future Generation Computer Systems* 27.6 (2011), pp. 743–756 (cit. on p. 27).

[68] K. Morik and M. Scholz. "The MiningMart Approach". In: *Informatik bewegt: Informatik*. 2002, pp. 811–818 (cit. on pp. 25, 111).

[69] M. A. Munson. "A Study on the Importance of and Time Spent on Different Modeling Steps". In: *ACM SIGKDD Exploration Newsletter* 13.2 (2012), pp. 65–71 (cit. on pp. 3, 62, 133).

[70] P. Nguyen, M. Hilario, and A. Kalousis. "Using Meta-mining to Support Data Mining Workflow Planning and Optimization". In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 605–644 (cit. on pp. 3, 7, 23, 117).

[71] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science". In: *Proceedings of the International Conference on Genetic and Evolutionary Computation Conference*. 2016, pp. 485–492 (cit. on pp. 3, 111).

[72] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil. "Improved Dataset Characterisation for Meta-learning". In: *Proceedings of the International Conference on Discovery Science*. 2002, pp. 141–152 (cit. on pp. 37, 59, 67).

[73] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier. "Meta-Learning by Landmarking Various Learning Algorithms". In: *Proceedings of the International Conference on Machine Learning*. 2000, pp. 743–750 (cit. on pp. 37, 59, 67).

[74] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999 (cit. on p. 64).

[75] J. Raes. "Inside two commercially available statistical expert systems". In: *Statistics and Computing* 2.2 (1992), pp. 55–62 (cit. on pp. 25, 110).

[76] V. Raman and J. M. Hellerstein. "Potter's Wheel: An Interactive Data Cleaning System". In: *Proceedings of the International Conference on Very Large Data Bases*. 2001, pp. 381–390 (cit. on pp. 2, 109).

[77] M. Reif, F. Shafait, and A. Dengel. "Meta2-Features: Providing Meta-Learners More Information". In: *Proceedings of the German Conference on Artificial Intelligence*. 2012 (cit. on p. 40).

[78] M. Reif, F. Shafait, and A. Dengel. "Meta-learning for evolutionary parameter optimization of classifiers". In: *Machine Learning* 87.3 (2012), pp. 357–380 (cit. on p. 7).

[79] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel. "Automatic Classifier Selection for Non-experts". In: *Pattern Analysis and Applications* 17.1 (Feb. 2014), pp. 83–96 (cit. on pp. 7, 35, 45, 59).

[80] L. Rendell, R. Seshu, and D. Tcheng. "Layered concept-learning and dynamically-variable bias management". In: *Proceedings of International Joint Conference on Artificial Intelligence*. 1987, pp. 308–314 (cit. on p. 58).

[81] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003 (cit. on p. 5).

[82] F. Serban, J. Vanschoren, J. Kietz, and A. Bernstein. "A survey of intelligent assistants for data analysis". In: *ACM Computing Surveys* 45.3 (2013), p. 31 (cit. on pp. 19, 23, 30, 34, 67).

[83] Y. L. Simmhan, B. Plale, and D. Gannon. "A Survey of Data Provenance in e-Science". In: *SIGMOD Record* 34.3 (2005), pp. 31–36 (cit. on p. 24).

[84] D. H. Sleeman, M. Rissakis, S. Craw, N. Graner, and S. Sharma. "Consultant-2: pre- and post-processing of ML applications". In: *International Journal of Human-Computer Studies* 43.1 (1995), pp. 43–63 (cit. on pp. 25, 110).

[85] S. Y. Sohn. "Meta Analysis of Classification Algorithms for Pattern Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.11 (1999), pp. 1137–1144 (cit. on p. 59).

[86] M. Sokolova and G. Lapalme. "A Systematic Analysis of Performance Measures for Classification Tasks". In: *Information Processing and Management* 45.4 (July 2009), pp. 427–437 (cit. on p. 71).

[87] H. Steck. "Evaluation of Recommendations: Rating-prediction and Ranking". In: *Proceedings of the ACM Conference on Recommender Systems*. 2013, pp. 213–220 (cit. on p. 107).

[88] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. "Data Curation at Scale: The Data Tamer System". In: *Online Proceedings of the Biennial Conference on Innovative Data Systems Research*. 2013 (cit. on p. 110).

[89] C. Thornton, F. Hutter, H. H. Hoos, et al. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. Chicago, Illinois, USA, 2013, pp. 847–855 (cit. on pp. 3, 80, 111).

[90] L. Todorovski, P. Brazdil, and C. Soares. "Report on the Experiments with Feature Selection in Meta-Level Learning". In: *Proceedings of the PKDD Workshop on Data Mining*. 2000, pp. 27–39 (cit. on p. 59).

[91] L. Todorovski and S. Dzeroski. "Experiments in Meta-level Learning with ILP". In: *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. 1999, pp. 98–106 (cit. on p. 59).

[92] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. "OpenML: Networked Science in Machine Learning". In: *ACM SIGKDD Explorations Newsletter* 15.2 (2014), pp. 49–60 (cit. on pp. 2, 35, 69, 87, 92).

[93] J. Varga, O. Romero, T. B. Pedersen, and C. Thomsen. "Towards Next Generation BI Systems: The Analytical Metadata Challenge". In: *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*. 2014, pp. 89–101 (cit. on pp. 25, 26, 29).

[94] D. H. Wolpert. "The Lack of a Priori Distinctions Between Learning Algorithms". In: *Neural Computation* 8.7 (1996), pp. 1341–1390 (cit. on p. 11).

[95] M. Záková, P. Kremen, F. Zelezný, and N. Lavrac. "Automating KD Workflow Composition Through Ontology-Based Planning". In: *IEEE Transactions on Automation Science and Engineering* 8.2 (2011), pp. 253–264 (cit. on pp. 25, 111).

Appendices

A

PRESISTANT: Data Pre-processing Assistant

A concrete classification algorithm may perform differently on datasets with different characteristics, e.g., it might perform better on a dataset with continuous attributes rather than with categorical attributes, or the other way around. Typically, in order to improve the results, datasets need to be pre-processed. Taking into account all the possible pre-processing operators, there exists a staggeringly large number of alternatives and non-experienced users become overwhelmed. Furthermore, trial and error is not feasible in the presence of big amounts of data.

We developed a method and tool — PRESISTANT, with the aim of answering the need for user assistance during data pre-processing. Leveraging ideas from meta-learning, PRESISTANT is capable of assisting the user by recommending pre-processing operators that ultimately improve the quality of the data analysis (e.g., increase the predictive accuracy of a classification algorithm). The user selects a classification algorithm and then PRESISTANT proposes candidate transformations to improve the result of the analysis.

In the demonstration, participants will experience, at first hand, how PRESISTANT easily and effectively ranks the pre-processing operators.

A.1 Introduction

The main tools used for data analysis (e.g., R[1], scikit-learn[2], Weka [38]) overlook data pre-processing when it comes to assisting non-expert users on improving the overall performance of their analysis. These tools are usually meant for professional users — who know exactly which pre-processing operators (transformations) to apply, and they leave non-experts unattended. To remedy this, we developed PRESISTANT[3] (cf. Figure A.1), which focuses on assisting the users by reducing the number of pre-processing options to a bunch of potentially relevant ones and ranks them. The goal is to highlight the transformations that have higher potential positive impact on the analysis.

PRESISTANT aims at reducing the time consumed in data pre-processing and at the same time improving the final result of the analysis. The focus is on classification problems, thus only operators that improve the performance of a classification algorithm (e.g., increase the predictive accuracy) are recommended.

Outline. We first give a brief overview of data pre-processing. Next, we give an overview of PRESISTANT and present its core features. Finally, we outline an on-site presentation.

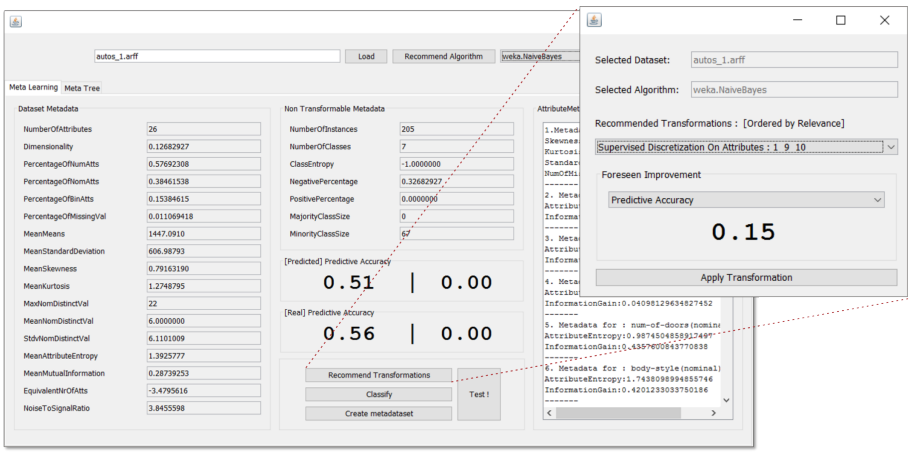


Fig. A.1: PRESISTANT: application interface

[1]https://r-project.org
[2]http://scikit-learn.org/stable
[3]For details visit: `http://www.essi.upc.edu/~bbilalli/presistant.html`

A.2 Data Pre-processing

Data pre-processing consumes 50-80% of data analysis time [69]. The reason for this is that it encompasses a broad range of activities. Sometimes data needs to be transformed in order to fit the input requirements of the machine learning algorithm, e.g., if the algorithm accepts only data of numeric type, data is transformed accordingly [49]. Sometimes, data requires to be transformed from one representation to another, e.g., from an image (pixel) representation to a matrix (feature) representation [32], or data may even require to be integrated with other data to be suitable for exploration and analysis [60]. Finally, and more importantly, data may need to be transformed with the only goal of improving the performance of a machine learning algorithm [17]. The first two types of transformations are more of a necessity, whereas the latter is more of a choice, and since an abundant number of choices exist, it is time consuming to find the right one. In PRESISTANT, we target the latter type of pre-processing, and as such, the transformations taken into consideration are of the type that can impact the performance of data mining algorithms (i.e., classification algorithms), and they are listed in Table 4.1. They are the most commonly used transformations in the Weka platform, and their implementations are open source[4]. The list of transformations considered by PRESISTANT can be extended by simply adding new ones to the palette.

A.3 System Overview

PRESISTANT takes as input a dataset and a classification algorithm, and generates a ranking of transformations according to their predicted impact on the final result of the algorithm. In the following, we explain the method and architecture used to achieve this.

A.3.1 Meta-learning for Data pre-processing

Meta-learning is a general process used for predicting the performance of an algorithm on a given dataset. It is a method that aims at finding relationships between dataset characteristics and data mining algorithms [46]. Given the characteristics of a dataset, a predictive meta-model can be used to foresee the performance of a given data mining algorithm. For instance, in a classifica-

[4]https://github.com/bnjmn/weka

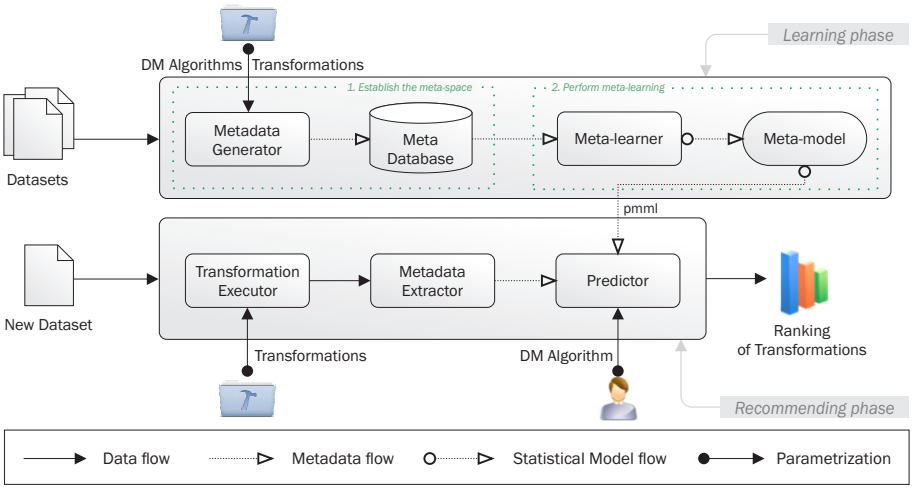


Fig. A.2: PRESISTANT: system overview

tion problem, meta-learning can be used to predict the predictive accuracy of a classification algorithm on a given dataset and hence provide user support in the mining step.

However, in Chapter 4 and 5, we showed that meta-learning can also be used to provide support in the pre-processing step. This can be done by learning the impact of data pre-processing operators (transformations) on the final result of the analysis. That is to say, detecting the transformations that after being applied on a dataset, increase the accuracy of a given classification algorithm on that dataset. This way, meta-learning pushes the user support to the data pre-processing step by enabling a ranking of transformations according to their relevance to the analysis.

Extensive results of the evaluation of this method can be found in Chapter 5 and Appendix A. Yet briefly, the evaluation of the rankings with regards to the gain obtained from the user's point of view, using a classical information retrieval metric — Discounted Cumulative Gain (DCG) [43], showed that for the whole set of possible transformations of a dataset, PRESISTANT is as close as 73% to the gain obtained from the ideal rankings, whereas for the best transformation in the ranking, PRESISTANT is as close as 79% on average for all the considered algorithms.

A.3.2 Architecture & Implementation

PRESISTANT is built with the meta-learning concept in mind and, as a consequence, it consists of two main phases, the *learning* and the *recommending* phase. Figure A.2 depicts the architecture of PRESISTANT.

The *learning* phase is performed offline and consists of two steps. In the first step, a meta-space is established for each classification algorithm that is considered by the system for application, i.e., for the time being PRESISTANT supports 5 classification algorithms. The meta-space is constructed by extracting dataset characteristics — meta-features, and by generating different measures on the performance of the classification algorithms on the datasets, i.e., predictive accuracy, precision, recall, and area under the roc curve (AUC). Dataset characteristics and performance measure altogether are referred to as *metadata*. In the second step, meta-learning is performed on top of the meta-space (meta-dataset). As a result, a predictive meta-model is generated that can be used to predict the performance of a classification algorithm on any new dataset.

The *recommending* phase is initiated when a new dataset to be analyzed arrives. At this point, a set of transformations are applied, and the corresponding transformed datasets are obtained. Transformations are applied depending on whether they are Local or Global (as classified in Table 4.1). If a transformation is Global it is applied only once to the set of all compatible attributes (e.g., normalizing all numeric attributes), whereas if it is Local, it is applied to: 1) every compatible attribute separately (e.g., discretizing one attribute at a time), and 2) all the set of compatible attributes (e.g., replacing missing values of all attributes). Furthermore, PRESISTANT uses heuristics (cf. Chapter 5 for more details), to prune the number of transformations that may be applied. For instance, given that *Standardization* and *Normalization* do not affect the performance of a *Decision Tree*, they are not applied when using a *Decision Tree (J48)* as classifier. Similarly, since in Weka, when applying *Nearest Neighbor (IBk)* the *Normalization* transformation is applied implicitly, PRESISTANT does not check for its impact in an explicit way. Although it may seem computationally costly to execute transformations and then predict the performance of algorithms on the transformed datasets, notice that this is orders of magnitude less costly than applying the classification algorithm after each transformation. This in fact, is the strongest point of our approach.

Furthermore, some concepts that require specific attention are:

Metadata. In Chapter 2, we studied and classified all types of metadata that can be used by systems that intelligently support the user during the process of data analysis. PRESISTANT, considers: 1) 54 dataset characteristics consisting of different summary characteristics (e.g., number of instances, dimensionality, class entropy, mean attribute entropy, etc.) and 2) a measure of the performance of an algorithm on the dataset (i.e., predictive accuracy, precision, recall or AUC). The metadata generation and extraction are performed in the *Metadata Generator* and *Metadata Extractor* modules, respectively, shown in Figure A.2. These modules are implemented in *Java*.

Meta-learner. The meta-learner is the algorithm that performs learning on top of the meta-dataset constructed out of the above mentioned metadata. The goal of the meta-learner is to create a model that is capable of predicting the performance of a given classification algorithm on a transformed dataset. In other words, its goal is to correctly predict the impact of a transformation on the performance of a classification algorithm. PRESISTANT uses a *Random Forest* as meta-learner. The meta-learning process is performed in the *Meta-Learner* module shown in Figure A.2, and it is implemented in *R*. The models generated are exported into *pmml* files, which are then fed to the *Predictor* module.

Transformations. The set of transformations currently considered in PRESISTANT, that the audience will experience, are described in Table 4.1 and cover a wide range of data pre-processing tasks, which are commonly used in the Weka platform. The execution of transformations is performed in the *Transformations Executor* module, shown in Figure A.2, which is implemented in *Java*.

Classification Algorithms. They represent the classification algorithms for which PRESISTANT can currently provide user support. That is, if the user selects one of these algorithms for analyzing a dataset, PRESISTANT is capable of recommending transformations that will improve the quality of the analysis. PRESISTANT is built on top of Weka, and Weka categorizes the classification algorithms into the following 7 categories: *bayes, functions, lazy, rules, trees, meta-methods and miscellaneous*, out of which the last two contain algorithms that are more complex and almost never used by non-experts. Assistants will be able to experiment with a representative classification algorithm for each category

except the last two, and they are: *Naive Bayes, Logistic, IBk, PART, and J48*, respectively.

A.4 Demo Walkthrough

Different real world datasets covering a variety of domains (e.g., health, insurance, banking) are available to show the benefits of our tool. Demo participants are encouraged to play the roles of data analysts and validate the tools' assistance. Since, for the time being, PRESISTANT covers 5 classification algorithms, the on-site demonstration consists of 5 scenarios where each time a different algorithm is presented. In every scenario, the user deals with a classification problem, e.g., in the first scenario the user is presented with the *lung cancer*[5] dataset where the task is to build a prediction model that achieves a good accuracy on predicting the type of the lung cancer a patient has, evaluated with 10-fold cross-validation. The idea is to show how the transformations recommended by PRESISTANT improve the quality of the analysis or more precisely how once applied, they increase the predictive accuracy of a chosen classification algorithm. Moreover, we validate the performance of PRESISTANT by comparing how classification algorithms perform on datasets without any transformations applied and how they perform on datasets transformed according to the recommendations of PRESISTANT. Further datasets for the on-site demonstration can be selected from *OpenML*[6], which contains one of the biggest collection of datasets for classification problems.

[5]https://www.openml.org/d/163
[6]http://www.openml.org

B

Evaluation Results for PRESISTANT

B.1 Evaluation Results

In the following we show the results obtained when evaluating 1) the quality (i.e., accuracy) of the top-K recommendations provided by PRESISTANT, and 2) the evaluation results (i.e., significance values) obtained when comparing the recommendations of PRESISTANT to a random pick. We show the results obtained for 4 classification algorithms such as: *J48 (Decision Tree)*, *Naive Bayes*, *PART*, and *Logistic*.

In particular, the blue cells show the accuracy values obtained for the corresponding classifiers, where each row denotes the position K of the transformations recommended and the columns denote the number of datasets with L real positive transformations. The numbers shown inside the cells, denoted as x/z, are the cumulative accuracies x, and the number of datasets z, which have at least K transformations and exactly L real positives. The last column shows the values obtained after computing the weighted average for each position K for all possible L.

Furthermore, the red cells show the significance values obtained when comparing the results obtained with PRESISTANT and the random pick for the corresponding classifiers. The numbers shown inside the cells, denoted as x/v, are the cumulative accuracies x obtained using our approach, and the

random pick probabilities z, for datasets with at least K transformations and exactly L real positives. The last column shows the averages for our approach compared to the random pick, in each position K, weighted by the number of datasets in each L.

In summary, observing the accuracy and significance values obtained, especially with regards to the top positions K, we can claim that PRESISTANT is capable of learning the impact of transformations on the classification accuracy.

Table B.1: Accuracy values followed by Significance values for the J48 (Decision Tree) classifier

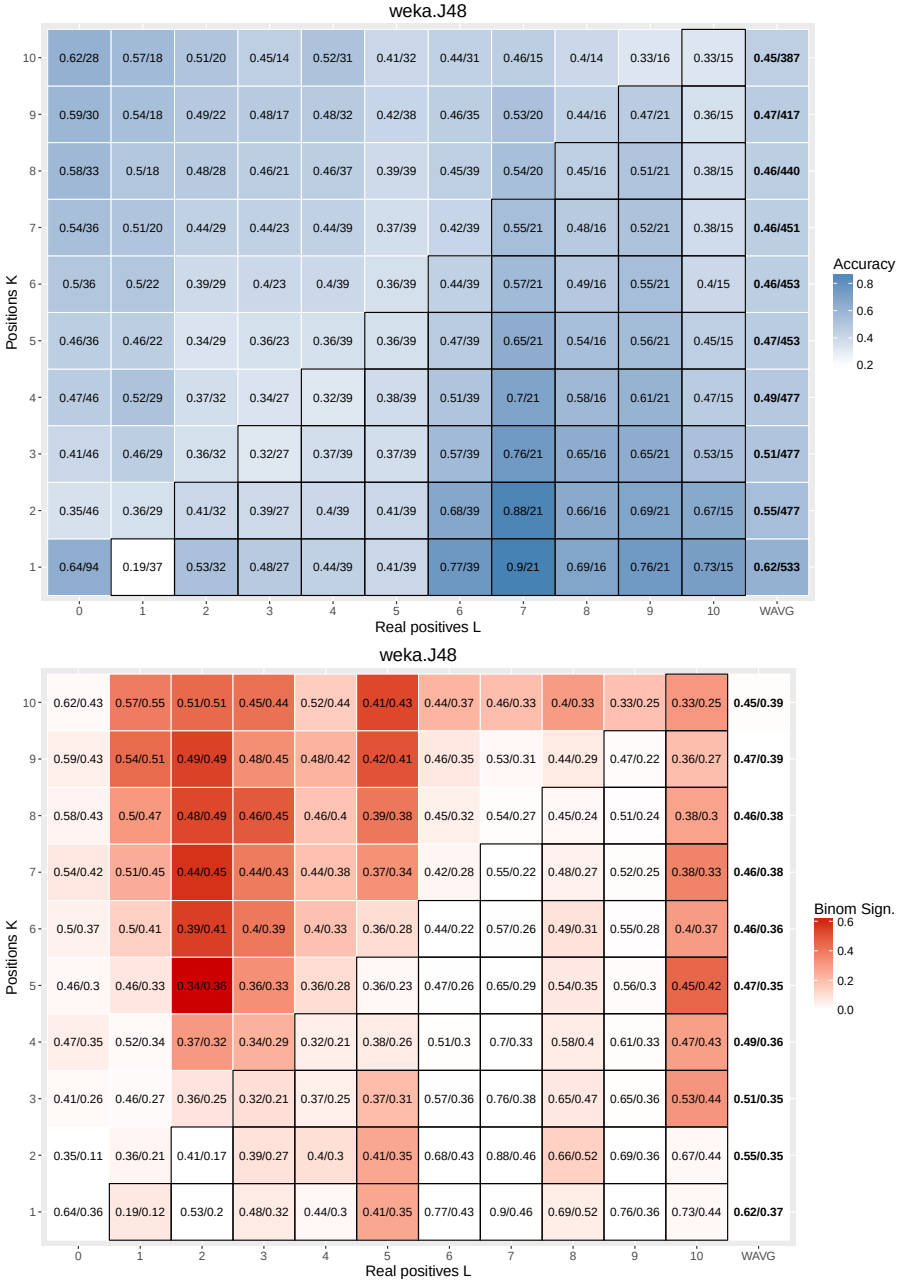


Table B.2: Accuracy values followed by Significance values for the NaiveBayes classifier

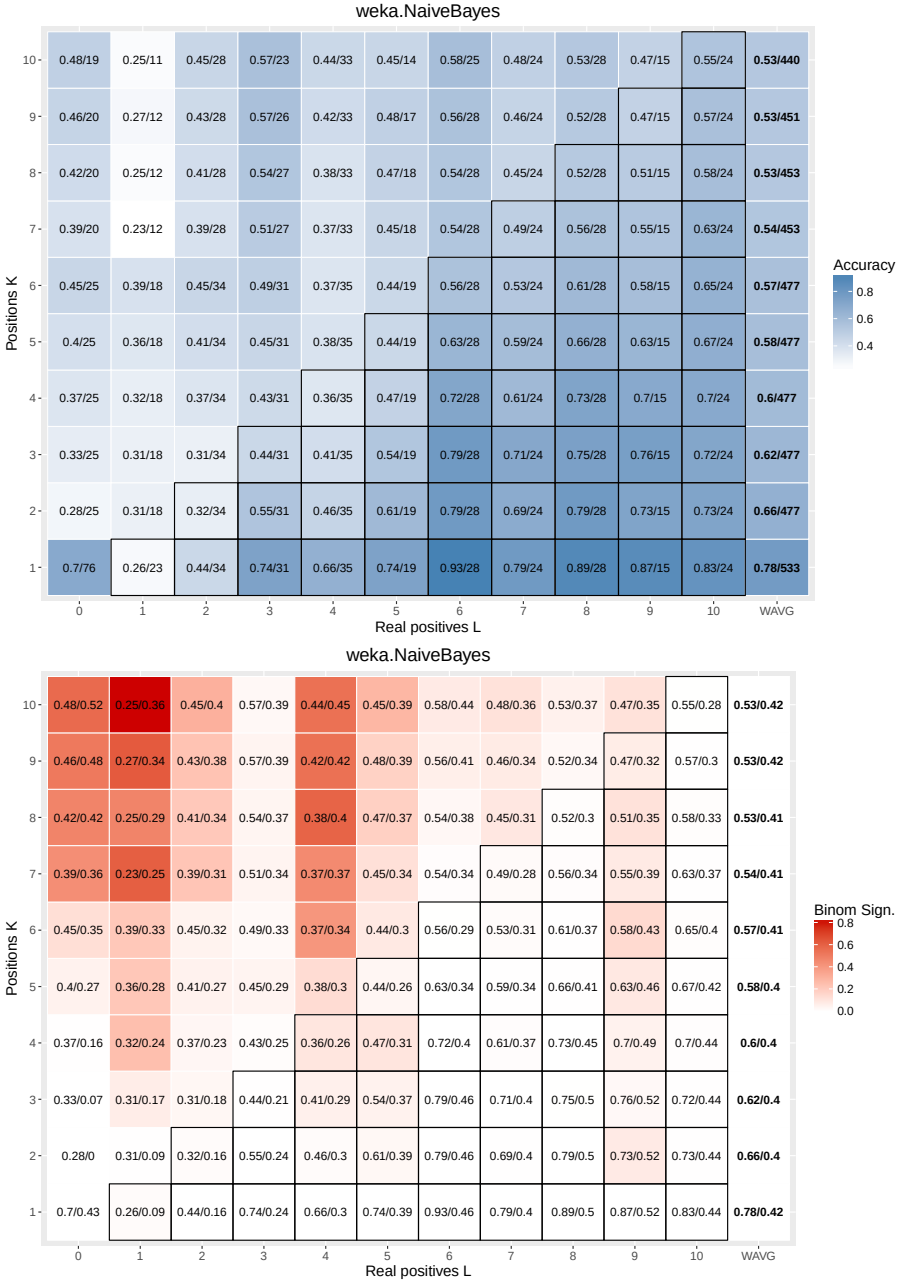


Table B.3: Accuracy values followed by Significance values for the PART classifier

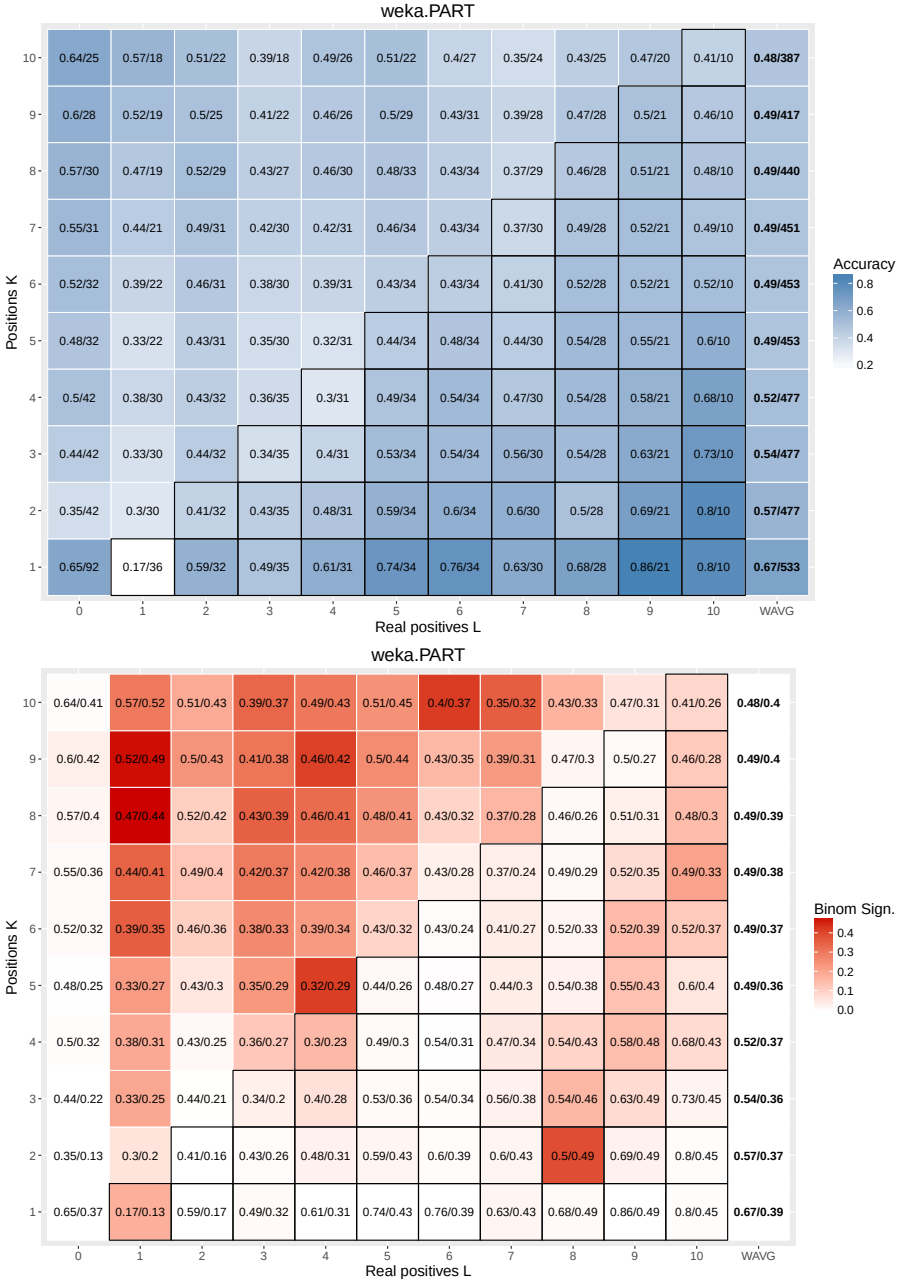


Table B.4: Accuracy values followed by Significance values for the Logistic classifier