Automated Data Pre-processing via Meta-learning

Besim Bilalli[1], Alberto Abelló[1], Tomàs Aluja-Banet[1], and Robert Wrembel[2]

[1] Universitat Politécnica de Catalunya, Barcelona, Spain
{bbilalli,aabello}@essi.upc.edu
tomas.aluja@upc.edu
[2] Poznan University of Technology, Poznan, Poland
robert.wrembel@cs.put.poznan.pl

Abstract. 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. As a matter of fact, a dataset usually needs 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. We show that this problem can be addressed by an automated approach, leveraging ideas from meta-learning. Specifically, we consider a wide range of data pre-processing techniques and a set of data mining algorithms. For each data mining algorithm and selected dataset, we are able to predict the transformations that improve the result of the algorithm on the respective dataset. Our approach will help non-expert users to more effectively identify the transformations appropriate to their applications, and hence to achieve improved results.

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 itself, a.k.a. knowledge discovery, consists of several steps, such as *data selection*, *data pre-processing*, *data mining*, and *evaluation* or *interpretation* [5], see Figure 1. One of the most important steps of this process is the data pre-processing step. Data pre-processing is so important that usually 50-80% of analysis time is spent on it [13]. The reason 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) [3].

Now, if the data pre-processing is so important and in addition it needs to be performed by a non-expert user, then a way must be found such that pre-processing becomes easy, i.e., offer assistance to the users in order to perform this step in a successful way.

In this paper, we propose a solution to this problem. We aim at assisting the user by recommending transformations i.e., pre-processing operators, that will

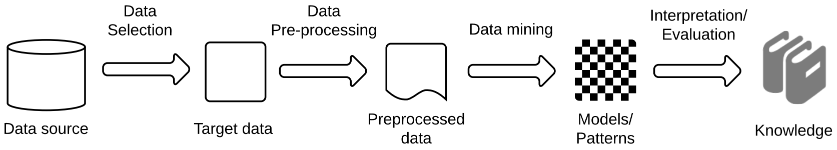

Fig. 1. Data Analysis/Knowledge Discovery process, adapted from [5].

ultimately improve the result of the analysis, that usually happens to be a prediction 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 data mining algorithm we are able to predict transformations that once applied yield an improved algorithm performance (e.g., predictive accuracy).

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

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

The rest of the paper is organized as follows: the related work is discussed in Section 2. An overview of data pre-processing, together with its benefits is given in Section 3. Our proposed solution is formally defined and a brief look at the implementation prototype is given in Section 4. The results of the experimental evaluations are reported in Section 5. Finally, Section 6 summarizes our work and outlines some future work.

2 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 [7], an open source tool for data mining, allows users to apply pre-processing 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 [16], 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 parametrisation without considering the pre-processing step. Hence, the user needs to deal with the pre-processing on his own.

In AmazonML[3], 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, however, are 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 [9], which is a product of the eLico[4] project, aims to autonomously construct 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 [12], since 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, hence, the data mining algorithm can be chosen at will.

There exist some other systems [2, 8, 11], however, they are also focused on providing support for the data mining step only.

3 Overview

3.1 Data Pre-processing

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, artificial neural networks, suport vector machines) capable of mining large datasets, 2) challenges in transforming attributes of different scales into mathematically feasible and computationally suitable formats [3]. Indeed, each attribute may require special treatment, such as discretization of

[3] https://aws.amazon.com/machine-learning

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

numerical attributes, rescaling of ordinal attributes and encoding of categorical ones. Hence, different transformations may be required.

For the sake of this paper, we consider the transformations shown in Table 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*. *Data reduction* aiming at decreasing the size of the dataset (e.g., instances selection or feature selection) and *data projection*, altering the representation of the data (e.g., mapping continuous values to categories or encoding nominal attributes) [14].

In Table 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 and `Local` — applied to specific compatible attributes, 3) the *Input Type*, which denotes the compatible attribute type for a given transformation, which can be `Continuous` — represent measurements on some continuous scale, or `Categorical` — represent 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`.

Transformation	**Technique**	**Attributes**	**Input Type**	**Output Type**
Discretization	Supervised	Global	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	Local	Continuous	Continuous
Replace Miss. Val.	Unsupervised	Global	Continuous	Continuous
Replace Miss. Val.	Unsupervised	Global	Categorical	Categorical
Principal Components	Unsupervised	Global	Continuous	Continuous

Table 1. List of transformations.

3.2 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 [3, 4].

Let us suppose that a user wants to apply the `Logistic` algorithm to the `Automobile`[5] dataset. The summary of `Automobile` is given in Table 2. This

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

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

Table 2. Summary of `autos`.

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

Table 3. Transformations on `autos`.

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` is applied to the original non-transformed dataset, a predictive accuracy of `0.71` is obtained with a 10 fold cross-validation. Note that for this run the Weka implementation of `Logistic` with 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 3 are obtained. In Table 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 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.

4 Our Solution

4.1 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. However, taking into consideration the above mentioned scenario where a user needs to be provided with some transformations to be applied, meta-learning can also be used to find relationships between transformations and data mining algorithms. That is because transformations by nature modify a dataset, and in turn, the dataset characteristics. As a matter of fact, transformations, through the changes they cause in the dataset characteristics, can be indirectly linked to a data mining algorithm. Hence, we can find/learn the relationships between

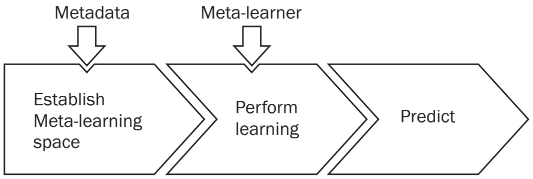

Fig. 2. Tree phases of the ranking process.

transformations and data mining algorithms. Thus, we use meta-learning to 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 2. First, a *meta-learning space* is established using metadata consisting of dataset characteristics along with some performance measures for data mining algorithms on those particular datasets, see Table 4. Then, the *meta-learning phase* generates a model (i.e., predictive meta-model) which defines the area of competence of the data mining algorithm [8]. Finally, when a transformed dataset (i.e., a transformation was applied on the dataset) arrives, the metadata are extracted and fed to the predictive meta-model, which predicts the performance of the algorithm — given the characteristics of the transformed dataset, and ultimately provides a ranking of the transformations. This concludes the *prediction* phase.

Name	# Instances	# Atts.	# Miss. Val.	# Cat. Atts.	# Cont. Atts.	PA
autos	205	26	59	11	15	0.71
credit	690	16	9	6	4	0.85
iris	150	5	0	0	4	0.97
vote	435	17	392	16	0	0.96

Table 4. An example (sample) from a meta-learning space. All columns except the last one, denote dataset characteristics. The last column is the predictive accuracy obtained if `Logistic` was applied to the respective dataset. Sign # means ”number of”.

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.

4.1.1 Metadata. In our previous work [1], 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.) [15] and may use different metadata. When it comes to meta-learning however, only dataset characteristics and performance characteristics of algorithms runs (i.e., predictive accuracies) on those datasets are used as metadata to establish the meta-space (see Table 4). Hence briefly, metadata in this case are a set of structural characteristics (e.g., extracted features) — the number of instances, the number of attributes, predictive accuracies of the algorithms runs on datasets, precisions of the algorithms runs on datasets, etc. — that jointly represent the relationships of algorithms with datasets. Different meta-learning systems may use different characteristics of datasets and different performance measures of algorithms runs in the meta-space. The metadata used in our approach is shown in Table 5.

Note that hundreds of metadata could be used and there is no defined methodology to find the set that will yield the best results. Moreover, their extraction might be costly and a tradeoff must be made between the amount of metadata to be used and the accuracy that can be obtained using them in the meta-learning phase.

In order to determine the metadata to be used, we followed an empirical approach. That is, we experimented with different combinations of metadata. Our experiments showed that the metadata in Table 5 give a good tradeoff. Note that they happen to coincide at a rate of 53% with the metadata used in the literature [2].

In Table 5, we also show the *Importance* of each metadata. The *Importance* coefficients are computed after generating the models in the meta learning phase, and they denote how important a metadata is, for creating the model. The bigger the coefficient, the more important the attribute is. We noticed that some metadata are assigned value `0`, that is, they were not used at all by the meta-learner. As a matter of fact, we removed them and recreated the models. Furthermore, in Table 5 column *Modifiable* indicates whether the metadata is modifiable through the transformations we use, shown in Table 1. If metadata are not transformable, we do not use them in the meta-learning phase, because those metadata remain constant and they do not reflect the impact of transformations. Hence finally, in the meta-learning phase we use only the metadata that are indicated with a check mark in the column *Used* in Table 5. Note, also the last row in Table 5, i.e., predictive accuracy, is the metadata we use as the performance measure of the algorithm on a specific dataset. In the meta-learning phase, this metadata is the one that needs to be predicted (i.e., the response). Naturally, columns *Importance* and *Modifiable* are not applicable to it, because this measure is not subject to transformations (e.g., transformations do not modify it directly).

4.1.2 Meta-learner.

Having stored an algorithm performance characteristic (i.e., predictive accuracy) and a set of dataset characteristics, the goal is to predict the performance of an algorithm in 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 [10], which splits the training data into n parti-

Metadata	**Importance**	**Modifiable**	**Used**
Negative Percentage	0.519947929	No	×
Class Entropy	0.472033619	No	×
Majority Class Size	0.366513463	No	×
Number of Instances	0.327327764	Yes	✓
Positive Percentage	0.24615823	No	×
Dimensionality	0.147883677	Yes	✓
Minority Class Size	0.144647803	No	×
Equivalent Number of Attributes	0.140606534	Yes	✓
Number of Features	0.123255813	Yes	✓
Percentage of Numeric Attributes	0.091996975	No	✓
Number of Classes	0.090051421	No	×
Noise to Signal Ratio	0.089608376	Yes	✓
Mean Kurtosis of Numeric Attributes	0.08734816	Yes	✓
Mean Means of Numeric Attributes	0.071206736	Yes	✓
Mean Std. Dev. of Numeric Attributes	0.056879682	Yes	✓
Mean Mutual Information	0.046159738	Yes	✓
Max. Nominal Att. Distinct Values	0.042345917	Yes	✓
Std. Dev. Nominal Att Distinct Values	0.040555858	Yes	✓
Mean Nominal Att. Distinct Values	0.040086227	Yes	✓
Mean Skewness of Numeric Attributes	0.025735383	Yes	✓
Percentage of Nominal Attributes	0.023476599	Yes	✓
Mean Attribute Entropy	0.021198277	Yes	✓
Percentage of Binary Attributes	0.009063724	Yes	✓
Percentage of Missing Values	0.002302323	Yes	✓
Incomplete Instance Count	0	Yes	×
Number of Instances With Missing Values	0	Yes	×
Min. Nominal Att. Distinct Values	0	Yes	×
Predictive Accuracy	NA	NA	✓

Table 5. The list of Metadata.

tions $D^{(1)}_{valid}, ..., D^{(n)}_{valid}$, 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_1 is the predictive accuracy of algorithm A run on that particular dataset. Hence, $\mathbf{x}$ and y altogether are the extracted metadata.

The meta-learner we decided to use is a *regression tree*. Trees have many good properties, such as: they perform implicit feature selection, require little effort for data preparation, nonlinear relationships between variables do not affect their performance and they are easy to interpret and explain. Thus, we created a regression tree for each data mining algorithm or more precisely for each classification algorithm.

In particular, the classification algorithms for which we generated regression trees are representative algorithms for all, except three classes of algorithms in Weka. In Weka, the classification algorithms are classified into: *bayes, functions, lazy, rules, trees, meta-methods, multi-instance methods, and ensemble-methods.*

We aimed at considering one algorithm for each one of the first five classes, and they are: *Naive Bayes, Logistic, IBk, JRip, and J48* respectively. The last three classes were omitted due to the fact that they are more complex and are not commonly used by non experienced users.

4.2 Solution Prototype

The general architecture of the developed solution prototype is depicted in Figure 3. The solution's main processes, the **Learning** and **Recommending** are implemented independently of each other. Below we give detailed explanations for each one of them.

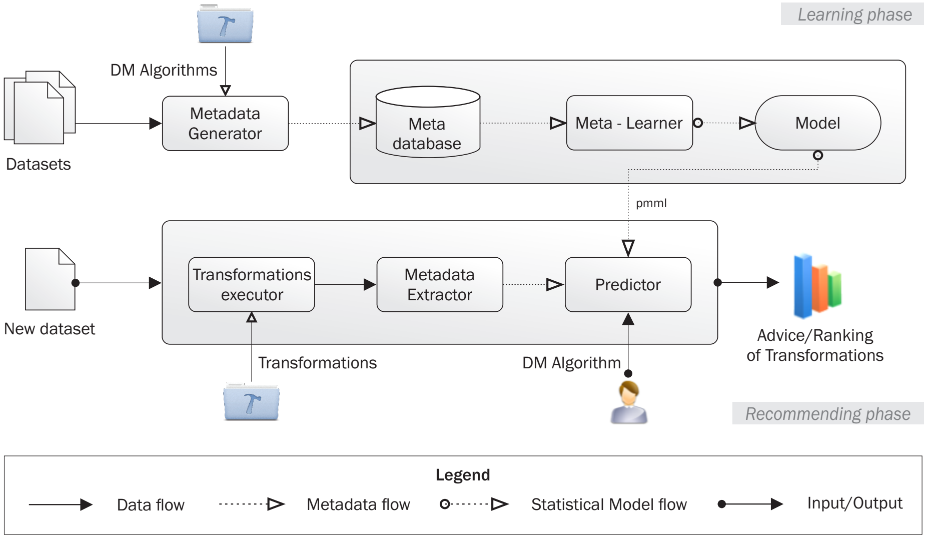

Fig. 3. Solution Architecture.

4.2.1 Learning phase. In the previous sections we mentioned that in order to build a model (e.g., predictive meta-model), we must firstly establish the meta-space — denoted as *Learning phase* in Figure 3. In our context, the meta-space needs to be constructed out of metadata that can be extracted from datasets and from the executions of algorithms on those datasets. As a matter of fact, we needed to fetch hundreds of datasets, extract some of their required characteristics, run different algorithms on them and get the predictive accuracies 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 [17] to fetch several hundred datasets. Next, from each dataset we extracted the 17 dataset characteristics — highlighted with a check-mark in Table 5. Finally, we executed the classification algorithms (see Section 4.1.2), on the datasets, and extracted the predictive accuracy values after evaluating with 10-fold cross validation. For each data mining algorithm, we obtained a meta-dataset that was fed to the *Meta-database*. In Figure 3, this is represented via the *Metadata creator* 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) through the *Meta-learners*, which in this case are regression trees. We used the `R` language to construct a tree for each one of the algorithms. Hence, we obtained one model per data mining algorithm. After that, the models were exported to `pmml` [6] 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. However, the models obtained are expected to slightly change from one collection to another.

4.2.2 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 metadata of the transformed dataset are extracted through the *Metadata extractor* module and they are fed to the *Predictor*, which using the model (pmml) of the respective algorithm (the one selected by the user) predicts the impact of the transformation. Finally, the Transformations are ranked according to their impact on the final result (according to whether they improve the final result). The modules of the *Recommending phase* are entirely developed in `Java`.

5 Evaluation

We performed an experimental study of the performance that can be achieved by our approach on various algorithms and various datasets. After specifying our experiment environment we evaluate our systems ability to predict the transformations that will improve the final result of the analysis.

5.1 Experimental setup

Recall that when building the meta-learners, we performed leaf-one-out validation for evaluating them (see Section 4.1.2). However, 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 tree using the subset of datasets (i.e., withholding the dataset that was left-out). Hence, for each data mining algorithm we created as many trees (meta-learners, meta-models) as datasets were used for training the tree of 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 tree (meta-model) that was built without using that particular dataset.

In our context, an experiment is performed in the following way. We first select a dataset and a data mining algorithm to be used for performing analysis (i.e., classification) on the dataset. Next, our system finds the **impact** of a **set of transformations** on the final result of the analysis.

The **set of transformations**, consists of iteratively applying the transformations shown in Table 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).

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

The *foreseen/predicted impact* is calculated by applying the set of transformations, as defined above, and subsequently extracting the characteristics (meta-data) of the transformed datasets, which are then used for predicting (foreseeing) the performance of the respective algorithm (on the transformed dataset).

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

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

5.2 Results for the improvements obtained by transformations

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 foreseen if the transformation were to be applied, *Bad* — a worsening of the final result for the respective algorithm is foreseen if the transformation were to be applied, or *Neutral* — neither improvement nor worsening is foreseen if the transformation were to be applied. Note that the latter occurs when the transformed dataset remains in the same leaf within the meta-learner (i.e., regression tree) or it moves to another leaf which predicts the same value (i.e., predictive accuracy). This is a limitation of the regression trees because they contain a discrete number of leaves, and hence a discrete number of possible predictions.

The aim of the experiments is to verify whether the foreseen categorizations are so for real. This, as previously mentioned (though costly) is done by executing

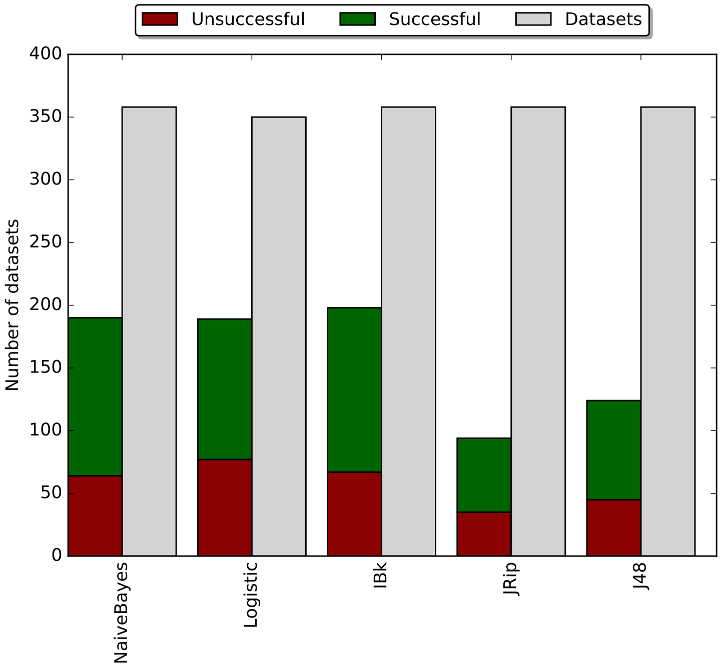

Fig. 4. Successful vs Unsuccessful Datasets.

the data mining algorithms on the transformed datasets and examining the real impact of the transformations.

In this context, we mark as *Successful*, the cases (i.e., datasets) on which the real average improvement we get from all the transformations categorized as *Good* for a dataset, is greater than the real average improvement we get from the transformations that were categorized as *Bad* for the same dataset. That is, the transformations foreseen as *Good*, "beat" on average the transformations foreseen as *Bad*.

In contrast, we mark as *Unsuccessful*, the cases on which the transformations foreseen as *Good* cannot "beat" on average the transformations foreseen as *Bad*.

In Figure 4, we show the results obtained. We show the comparison between the number of *Successful* cases — the green bar, and the number of *Unsuccessful* cases — the red bar. In addition, a third bar (i.e., gray) in the figure, denotes the total number of cases (datasets) for which we performed the experiments on each respective algorithm. Note 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 (54.7%), or we do not find *Bad* transformations (38.1%), or neither *Good* nor *Bad* transformations (7.2%). On the latter cases we could not find neither *Good* nor *Bad* transformations, but for the rest this happens because the datasets already belong to the best or the worst leaves of the trees (meta-learners), hence there can be no transformations that can move them 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 6. The column *p-val* 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 value in *p-val* is below or equal to `0.05`. As a matter of fact, we can observe that our method gives significant values for all the algorithms considered.

Algorithm	**Weka class**	**Successful**	**Successful+Unsuccessful**	**p-val**
Naive Bayes	Bayes	126	190	1.99359E-06
Logistic	Functions	112	189	0.004326558
IBk	Lazy	131	198	1.57706E-06
JRip	Rules	59	94	0.004773905
J48	Trees	79	124	0.000782367

Table 6. Binomial Significance Test for all Algorithms.

6 Conclusions and Future Work

In this work, we have shown that the daunting problem of data pre-processing can be alleviated by a practical, automated tool. This is 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 tool 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 a set of algorithms even blindly (e.g., users without any prior knowledge wrt data mining) applying the recommended transformations improves the final result of the algorithm. 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.

We see several promising avenues for future work. First, some limitations of using regression trees as meta-learners were observed (e.g., many transformations predicted to perform the same, due to the discrete number of leaves), suggesting the investigation of more sophisticated methods (e.g., neural networks). Second, we see potential value in customizing the transformations depending on the class of algorithms (e.g., trees) or even specific algorithms. Finally, we aim at extending the range of the classification algorithms that we have considered so far.

Acknowledgments. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC).

References

1. B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Towards intelligent data analysis: The metadata challenge. In *IoTBD*, 2016.
2. M. Charest et al. Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *IDA*, 2008.
3. S. F. Crone, S. Lessmann, and R. Stahlbock. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, 2006.
4. T. Dasu and T. Johnson. *Exploratory data mining and data cleaning*, volume 479. John Wiley & Sons, 2003.
5. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 1996.
6. A. Guazzelli, M. Zeller, W.-C. Lin, G. Williams, et al. Pmml: An open standard for sharing models. *The R Journal*, 2009.
7. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et al. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 2009.
8. A. Kalousis and M. Hilario. Model selection via meta-learning: A comparative study. *International Journal on Artificial Intelligence Tools*, 2001.
9. 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 *ESWC*, 2014.
10. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, 1995.
11. D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.
12. I. Mierswa. Rapid miner. *Künstliche Intelligenz*, 2009.
13. M. A. Munson. A study on the importance of and time spent on different modeling steps. *SIGKDD Explor. Newsl.*, 13(2), May 2012.
14. D. Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann, 1999.
15. F. Serban, J. Vanschoren, J. Kietz, and A. Bernstein. A survey of intelligent assistants for data analysis. *ACM Computing Surveys*, 2013.
16. C. Thornton, F. Hutter, H. H. Hoos, et al. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, 2013.
17. J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. Openml: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 2014.