PRESISTANT: Data Pre-processing Assistant

Chapter · June 2018

DOI: 10.1007/978-3-319-92901-9_6

5 authors, including:

Besim Bilalli
Universitat Politècnica de Catalunya
7 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Alberto Abelló
Universitat Politècnica de Catalunya
114 PUBLICATIONS **1,526** CITATIONS
SEE PROFILE

Tomas Aluja-Banet
Universitat Politècnica de Catalunya
55 PUBLICATIONS **217** CITATIONS
SEE PROFILE

Rana Faisal
Technische Universität Dresden
11 PUBLICATIONS **27** CITATIONS
SEE PROFILE

PRESISTANT: Data Pre-processing Assistant

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

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

Abstract. 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. 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 classification performance. The user selects a classification algorithm, from the ones considered, 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.

Keywords: Data pre-processing; meta-learning; data mining

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 being included in many Information Systems, and 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 to perform data analytics. However, the multiple steps involved in the data analytics process render this process challenging.

Data analytics, sometimes referred to as knowledge discovery [8], 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.

Table 1. Summary of `Automobile`

Metadata	Value
Instances	205
Attributes	26
Classes	2
Categorical Attributes	11
Continuous Attributes	15
Missing Values	59

Table 2. Transformations on `Automobile`

Transformation	Attribute	PA
Unsup. Discretization	1,9,10,11,12,13	**0.81**
Unsup. Discretization	1,9,10	**0.80**
Unsup. Discretization	All Cont. Atts.	**0.75**
Sup. Nom. To Binary	All Cat. Atts.	**0.73**
Unsup. Normalization	All Cont. Atts.	**0.71**

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 [16], where performance is measured in terms of the ratio of correctly classified instances (i.e., predictive accuracy). A brief real-world example can be used to demonstrate this.

Let us suppose that a multinational insurance company has an Information System to be used by different actuaries world wide. These want to apply a classification algorithm (e.g., Logistic Regression) to different *Automobile*[3] datasets like the one, whose summary is given in Table 1. 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 it can take the following values: $-3, -2, -1, 0, 1, 2, 3$ (value 3 indicates that the auto is risky, -3 that it is pretty safe). The goal 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 a 10 fold cross-validation. In contrast, if some pre-processing were to be applied in advance, the results shown in Table 2 would be obtained, where 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 Regression 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 predictive accuracy. A non-experienced user (at times even an experienced user) cannot be aware of that. His/her only solution is to execute all the possible transformations and then apply the algorithm after each transformation. However, this is generally unfeasible in the presence of big amounts of data, due to the high computational complexity of data mining algorithms. Hence, a proper recommendation of transformations would ease the user's task and at the same time it would improve the final result.

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

Table 3. 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

The main tools used for data analytics (e.g., R[4], scikit-learn[5], Weka [11]) 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[6], 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.

The remainder of this paper is organized as follows. 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 our on-site presentation.

2 Data Pre-processing

Data pre-processing consumes 50-80% of data analysis time [17]. 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 [14]. Sometimes, data requires to be transformed from one representation to another, e.g., from an image (pixel) representation to a matrix (feature) representation [10], or data may even require to be integrated with other data to be suitable for exploration and analysis [15]. Finally and more importantly, data may need to be transformed with the only goal of improving

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

the performance of a machine learning algorithm [7]. 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 classification algorithms, and they are listed in Table 3. These are the most commonly used transformations in the Weka platform, and their implementations are open source[7].

In Table 3, 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* — 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*.

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.

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 [6]. Given the characteristics of a dataset, a predictive meta-model can be used to foresee the performance of a given data mining algorithm. To this end, meta-learning has shown to be effective in the model-selection problem [1,13] as well as CASH (combined algorithm selection and hyperparameter optimization) problem [9,18].

However, in our previous works [2,4,5], we showed that meta-learning can also be used to provide support specifically in the pre-processing step. This can be done by learning the impact of data pre-processing operators 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 selected classification algorithm. 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.

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

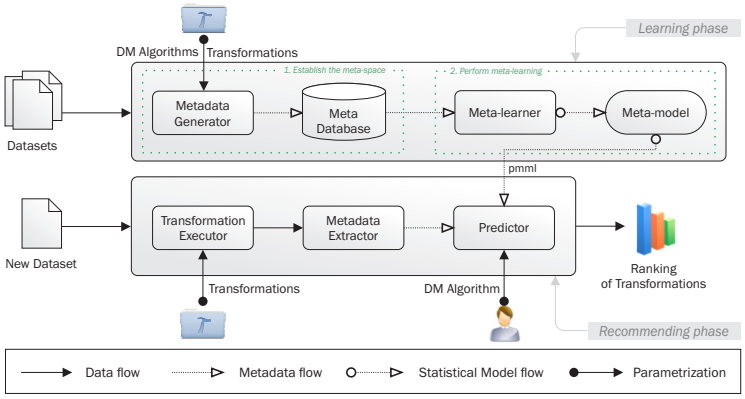

Fig. 1. PRESISTANT: system overview

Extensive results of the evaluation of this method can be found in [5]. 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) [12], 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 [5].

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 1 depicts the architecture of PRESISTANT.

The *learning* phase is performed offline and consists of two steps. In the first step, a meta-dataset 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-dataset 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-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 3). If a transformation is Global it is applied only once to the set of all compatible attributes (e.g.,

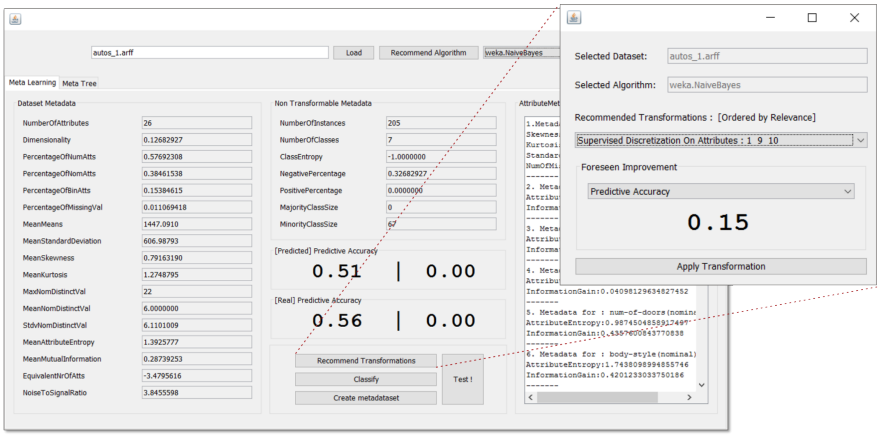

Fig. 2. PRESISTANT: application interface

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. [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 our previous work [3], we studied and classified all types of metadata that can be used by systems that intelligently support the user in the different steps of the data analytics process. 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 1. 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 1, and it is implemented in *R*. The generated models 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 3 and cover a wide range of data pre-processing tasks, which are distinguished as *data reduction* and *data projection*. *Data reduction* aims at decreasing the size of the dataset (e.g., instance selection or feature extraction). *Data projection*, alters the representation of the data (e.g., mapping continuous values to categories or encoding nominal attributes). The list of transformations considered by PRESISTANT can be extended by adding new ones from the palette of transformations offered in Weka, and then training the meta-models. The execution of transformations is performed in the *Transformations Executor* module, shown in Figure 1, which is implemented in *Java*.

Classification algorithms. They represent the 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.

4 Demo Walkthrough

In the on-site demonstration, the functionalities and capabilities of PRESISTANT (cf. Figure 2) for assisting the user in the data pre-processing step will be presented. Different real world datasets covering a variety of domains (e.g., health, insurance, banking) will be available to show the benefits of our tool. Demo participants will be 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 will consist of 5 scenarios where each time a different algorithm will be presented. In every scenario, the user will deal with a classification problem, e.g., in the first scenario the user will be presented with the *lung cancer*[8] dataset where the task will be 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

[8] https://www.openml.org/d/163

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. Further datasets for the on-site demonstration will be selected from *OpenML*[9], which contains one of the biggest collection of datasets for classification problems.

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, and T. Aluja-Banet. On the predictive power of meta-features in openml. *Applied Mathematics and Computer Science*, 27(4):697–712, 2017.
2. B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Automated data preprocessing via meta-learning. MEDI '16, pages 194–208, 2016.
3. B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Towards intelligent data analysis: The metadata challenge. IOTBD '16, pages 331–338, 2016.
4. B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Intelligent assistance for data pre-processing. *Computer Standards & Interfaces*, 57:101–109, 2018.
5. B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. PRESISTANT: Learning based assistant for data pre-processing. In *eprint arXiv:* `https://arxiv.org/pdf/1803.01024.pdf`, 2018. (Under review).
6. P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1 edition, 2008.
7. X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. SIGMOD '16, pages 2201–2206, 2016.
8. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 1996.
9. M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. NIPS '15, pages 2962–2970, 2015.
10. T. Furche, G. Gottlob, L. Libkin, G. Orsi, and N. W. Paton. Data wrangling for big data: Challenges and opportunities. EDBT '16, pages 473–478, 2016.
11. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 2009.
12. K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. SIGIR '00, pages 41–48, 2000.
13. A. Kalousis. Algorithm selection via meta-learning. 2002. Ph.D. Dissertation.
14. S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. CHI '11, pages 3363–3372, 2011.
15. M. Lenzerini. Data integration: A theoretical perspective. PODS '02, pages 233–246, 2002.
16. D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. *Machine Learning: Neural and Statistical Classification*. Ellis Horwood, 1994.
17. M. A. Munson. A study on the importance of and time spent on different modeling steps. *ACM SIGKDD Exploration Newsletter*, 13(2):65–71, 2012.
18. P. Nguyen, M. Hilario, and A. Kalousis. Using meta-mining to support data mining workflow planning and optimization. *J. Artif. Intell. Res.*, 51:605–644, 2014.

[9] http://www.openml.org