On Usefulness of Outlier Elimination in Classification Tasks: Extended Abstract

Dušan Hetlerovič 445557@MAIL.MUNI.CZ
Luboš Popelínský POPEL@FI.MUNI.CZ
KD Lab, FI MU Brno, Czechia

Pavel Brazdil PBRAZDIL@INESCTEC.PT
LIAAD - INESC TEC / FEP, Univ. of Porto, Portugal

Carlos Soares CSOARES@FE.UP.PT
Fraunhofer Portugal AICOS / LIACC / FEUP, Univ. of Porto, Portugal

Fernando Freitas FMLFREITAS@GMAIL.COM
FEUP, Univ. of Porto, Porto, Portugal

This extended abstract is based on a published article with the same title and authors, which appeared in the Proceedings of the International Symposium on Intelligent Data Analysis (IDA-2022), pp. 143-156, 2022.

Introduction and Objectives

Although outlier detection/elimination has been studied before, few comprehensive studies exist on when exactly this technique would be useful as preprocessing in classification tasks. Our objective is identify the most useful workflows for a given set of tasks (datasets), and then examine which outlier elimination methods (OEMs) appear in these workflows. The workflows considered in this work are pipelines that include an outlier elimination step followed by a classifier. The OEMs identified this way are considered as useful. Our final aim is to verify what effect this alteration has on generalization performance.

Method for the reduction of portfolios of workflows

The reduction method used here is based on the method in Abdulrahman et al. (2019), but includes various adaptations. This method uses a given portfolio of algorithms (in general workflows) and reduces it by removing non-competitive ones, by exploiting the existing performance metadata obtained in prior tests. This is followed by the elimination of workflows that include infrequent OEMs.

Identifying the most competitive algorithm is done by identifying the $N\%$ of top workflows for a given dataset. In this work we used the top 1% of workflows based on A3R measure (combining accuracy and time) and another top 1% of workflows based on accuracy only. All these workflows are passed to the second phase, whose aim is to eliminate all workflows which include rather infrequent OEM variants. If a particular OEM variant appears in less than P% of workflows, the corresponding workflows with this variant are marked for elimination. After processing all OEM variants, all corresponding workflows are dropped.

Algorithm/workflow recommendation method used

In this study, we have chosen the method *average ranking (AR*)* (Abdulrahman et al., 2018), as the algorithm/workflow recommendation method. This method was chosen because it is relatively simple and, consequently, it is easy to define different configurations that include all required alternatives (selected classification algorithms with/without OEMs). We have excluded AutoWeka, Auto-sklearn or other systems from consideration, as they not include all the OEMs we have considered here. Method AR* requires that each portfolio of workflows is converted into a ranking on the basis of available past performance metadata. Each ranking is then followed to generate recommendations for the task dataset. This enables to obtain its performance and to calculate how far it is from the best possible performance, i.e., calculate the *loss*. The evaluation follows the leave-one-out (LOO) strategy, so this is repeated as many times as there are datasets and the mean/median loss returned.

Experimental setup

In this study we have considered 12 different outlier elimination methods (OEMs); six of those were general, other six class-based (the class attribute is principal for outlier detection), representing a richer set than the one used by Smith and Martinez (2018). The OEMs included also a hyperparameter (6 different settings), indicating the percentage of top outliers to be eliminated. The OEMs discussed above were used in conjunction with 10 different classification algorithms. So this resulted in 730 different workflows to consider. We have conducted experiments with these workflows on 50 different datasets from OpenML.

Results

Our results show that if we use OEMs, the results of the workflow recommendation system will improve on average. The gain observed on a study with 50 datasets in a leave-one-out mode was rather significant (0.5%). The reduction method has identified that just three OEMs that are generally useful for the given set of tasks, namely RF-OEX (Random Forest outlier detection based on (dis)similarity (Nezvalová et al., 2015), TDwP (tree depth with pruning method (Smith et al., 2014), and DS (outlyingness is based on the size of leaf node of instances in the decision tree (Sumner et al., 2005). Besides, we showed that it is possible to eliminate 86% of the original workflows and still maintain the same loss when testing different alternatives.

Acknowledgements

The authors are grateful for the support under National Funds through the Portuguese funding agency, FCT Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020 of INESC TEC.

References

S. M. Abdulrahman, P. Brazdil, J. N. van Rijn, and J. Vanschoren. Speeding up algorithm selection using average ranking and active testing by introducing runtime. *Machine Learning*, 107(1):79–108, Jan 2018. ISSN 1573-0565. doi: 10.1007/s10994-017-5687-8. URL https://doi.org/10.1007/s10994-017-5687-8.

SM Abdulrahman, P. Brazdil, Wan MNW Zainon, and A. Adamu. Simplifying the algorithm selection using reduction of rankings of classification algorithms. In *ICSCA '19 Proc. of the 2019 8th Int. Conf. on Software and Computer Applications*, pages 140–148. ACM New York, 2019.

L. Nezvalová, L. Popelínský, L. Torgo, and K. Vaculík. Class-based outlier detection: staying zombies or awaiting for resurrection? In *IDA 2015, Saint Etienne*, pages 193–204. Springer, 2015.

M. R. Smith and T. R. Martinez. The robustness of majority voting compared to filtering misclassified instances in supervised classification tasks. *Artif. Intell. Rev.*, 49(1):105–130, 2018. doi: 10.1007/s10462-016-9518-2. URL https://doi.org/10.1007/s10462-016-9518-2.

M. R. Smith, T. R. Martinez, and Ch. G. Giraud-Carrier. An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256, 2014. doi: 10.1007/s10994-013-5422-z. URL https://doi.org/10.1007/s10994-013-5422-z.

M. Sumner, E. Frank, and M. Hall. Speeding up logistic model tree induction. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 675–683. Springer, 2005.