On Usefulness of Outlier Elimination in Classification Tasks

Dušan Hetlerović[1], Luboš Popelínský[1], Pavel Brazdil[2](✉), Carlos Soares[3], and Fernando Freitas[4]

[1] KD Lab, FI MU, Brno, Czechia
{445557,popel}@fi.muni.cz
[2] LIAAD - INESC TEC/FEP, University of Porto, Porto, Portugal
pbrazdil@inesctec.pt
[3] Fraunhofer Portugal AICOS and LIACC, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
csoares@fe.up.pt
[4] FEUP, University of Porto, Porto, Portugal

Abstract. 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. The objective of our study is to fill in this gap. We have performed experiments with 12 various outlier elimination methods and 10 classification algorithms on 50 different datasets. The results were then processed by the proposed reduction method, whose aim is identify the most useful workflows for a given set of tasks (datasets). The reduction method has identified that just three OEMs that are generally useful for the given set of tasks. We have shown that the inclusion of these OEMs is indeed useful, as it leads to lower loss in accuracy and the difference is quite significant (0.5%) on average.

Keywords: Outlier elimination · Metalearning · Average ranking · Reduction of portfolios

1 Introduction

One of the common problems machine learning users face is choosing an algorithm for a specific task [23]. The motivation is to either maximize or minimize a quantifiable measure, such as predictive accuracy. Apart from algorithm selection, users may achieve an improvement in performance by incorporating different data preprocessing methods and by selecting appropriate hyperparameter settings for these components. The combination of these three factors constitutes a *workflow (pipeline)* design problem. The search space of alternatives is sometimes referred to as *configuration space*. As this space can be very large, we need tools to help us identify the optimal one for the new task.

Many approaches focus on the issue of how to conduct the search in the given configuration spaces [28]. Other approaches try to redesign this space first in order to facilitate the search to be conducted in the future. The work described here falls into the second category. The space considered here includes different machine learning algorithms and outlier elimination methods (OEMs), which can be seen as a particular preprocessing method [25].

Various authors have shown that using outlier elimination as a preprocessing method can improve accuracy [25,26], although this may not always be beneficial. We have decided to investigate this issue. Our first aim was to identify which OEMs are *potentially useful* when taking into account the given classification algorithm and dataset. Our results show that, for some algorithms, such as Naive Bayes, the improvement is quite significant. However, various workflows that include OEMs have the opposite effect, namely they decrease the overall performance. Therefore, we have decided to conduct a more thorough study to determine whether the OEMs are *truly useful* when recommending algorithms for new unseen datasets.

To determine this, we have adapted the approach of [2]. In this work, the authors examined the usefulness of different algorithms (workflows) in a given portfolio, while taking into account a given set of tasks. The algorithms (workflows) that are unlikely to lead to a overall performance improvement are dropped. So, the algorithms (workflows) that remain after the reduction are of interest.

This study has shown that only three OEMs out of the initial set of twelve OEMs are required for solving the given set of tasks. So, these three OEMs that have been identified represent a generally useful knowledge that can be exploited in the design of other, more complex, algorithm recommendation platforms.

2 Related Work

Outlier Detection End Elimination. One pioneering work in the area of outlier detection and elimination was the work of John and Langley (1995) [16] on the so-called robust decision trees, who studied the effects of label noise. After learning a tree, all misclassified instances were removed from the learning set and a new tree was learned. This was repeated until the learning set was consistent. Although it did not result in accuracy increase, the resulting tree was much smaller. A similar approach for kNN was presented elsewhere [29,33].

Smith and Martinez (2018) [25] explored filtering of misclassified instances. Misclassification was studied both in conjunction with a single classifier or an ensemble of classifiers. In total, 54 datasets were used in conjunction with 9 supervised learning algorithms from Weka [12]. In both cases, misclassified instances were removed. When the same learning algorithm was used to filter misclassified instances and to learn a model, only three algorithms displayed an accuracy increase – LWL lazy learner, Neural net and Ripper. In all cases, using an ensemble of learning algorithms for filtering resulted in a greater increase in classification accuracy than when using a single learning algorithm. However, if compared with majority voting ensemble of the 9 classifiers, the majority voting ensemble reached, on average, the highest accuracy. An extensive and recent discussion of noise filters, as well as a particular solution for elimination of attribute noise can be found in [24].

Workflow Recommendation with AutoML/Metalearning Systems. Various approaches exist regarding how to identify the best possible workflow/pipeline (sequence of algorithms) for a given task and for a given configuration space. Some simple ones include, for instance, grid search, random search, and gradient descent method, which is often used in the task of configuring the hyperparameters of neural networks. The approach known as *sequential model-based search/optimization (SMBO)* exploits knowledge of past experiments on the target dataset [14]. The so-called *surrogate model* permits to carry out a relatively fast test to estimate the next best candidate to test. The system AutoWeka [28], for instance, uses this kind of search to identify the potentially best workflow configuration for a given task.

Metalearning approaches gather test results on various datasets and the metaknowledge obtained is used to construct a model estimating the next best candidate to test [5,32]. The metaknowledge gathered represents a set of workflows (pipelines) used in the past, some of which may be useful for the new task. Each workflow (pipeline) can also be seen as a *plan* of different operations to execute. So, one advantage of the metalearning approaches is that one does not need to search for a new plan if a sufficiently rich set already exists. The method called *AR** [1] exploits an *average ranking* of workflows. It represents a simple method that orders the workflows according to a given performance measure (e.g., accuracy, a combined measure of accuracy and runtime). Hence, this method allows us to evaluate the benefit of adding new workflows to a given portfolio and thus obtain information about its *marginal contribution to performance* [34].

Analysis and Reformulation of Given Configuration Space. Various authors have investigated the issue of which parts of the given configuration space are useful for a given set of tasks. Various works exist on the problem of how to establish the relative importance of hyperparameters and their setting (see e.g., [30,34]). Others tried to use the results of prior analysis to reformulate the configuration space. As was mentioned earlier, [2] examined the usefulness of different algorithms (workflows) for a given set of tasks. The results of this analysis was used to reformulate the existing set (portofolio) of algorithms (workflows).

3 Research Questions and Methodology

Let us first list the main research questions that we wish to answer in this work:

RQ1: Can we use OEMs in workflows without restrictions?
RQ2: Are some OEMs potentially useful?
RQ3: Can we identify the most useful workflows with OEMs? If so, how?

The research question RQ1 is addressed in a study described in Sect. 5.1. The method used to answer the research question RQ2 is discussed in Sect. 3.2 and the results are presented in Sect. 5.2. The research question RQ3 is addressed in Sect. 3.3 and the results are presented in Sect. 5.3.

3.1 Basic Concepts

Let us clarify the meaning of some basic concepts, starting with the notion of *outliers*. Barnett and Lewis [4] define them as observations that deviate so much from the rest of the data that it is likely that they are generated by a different phenomenon than the one being analyzed. If we characterize the points generated by a certain distribution, then outliers can be seen as the points that do not belong to this distribution.

We can distinguish two kinds of noise in a dataset that may even influence each other [35]. *Class noise* appears when instances are incorrectly labelled and may be caused in a process of labelling by a human, while *attribute noise* corresponds to errors in attribute values - caused, for instance, by measurement errors. While class noise may be eliminated by instance filtering, for attribute noise it is not appropriate. The work presented here focuses on attribute noise.

Let us also clarify what we mean by initial and extended of workflows. The *initial workflows* are of the form CL_k represents a particular classification algorithm with default settings. The set of classifiers used in the experiments is shown in Sect. 4. The *initial portfolio* includes the set of these initial workflows.

The *extended workflows* are of the form $OEM_{i,j}, CL_k$, where $OEM_{i,j}$ represents the outlier elimination method i with configuration j. The set of outlier elimination methods (OEMs) used in the experiments is shown in Sect. 4. The extended portfolio includes both the initial and the extended workflows.

3.2 Determine Whether Some OEMs are Potentially Useful

Informally speaking, the extended workflow $(OEM_{i,j}, CL_k)$ can be considered to be *potentially useful* if it leads to increased performance on many datasets when compared to its initial counterpart (CL_k). The amount of the increase also matters and so we also take this into account. The aim of our experiments are twofold: first, determine whether all, or just some, of the extended workflows with OEMs can be considered as useful. If at least some extended workflows are identified as potentially useful, our aim is to identify the classification algorithm and the datasets involved. The results of this study are presented in Sect. 5.2.

3.3 Identify the Most Useful Workflows with OEMs

Our aim is to compare the performance of a chosen algorithm selection method on two different portfolios, the initial and the extended one, which may include some workflows with OEMs. The aim of this comparison is to determine whether it is advantageous to use the extended portfolio. However, we need to be careful, as some workflows that include an OEM may result in a decrease of performance. So, to avoid this, we use a reduction method based on [1] to select the most competitive workflows. This way, the extended set is effectively reduced, by pruning out the non-competitive variants. So one key question is the following: will some of the OEMs "survive" this reduction phase? If so, which ones? Will the final portfolio lead to competitive results? The aim of our experiments discussed in Sect. 5 is to answer these questions and, this way, shed light on the usefulness of outlier elimination methods.

Method for the Reduction of Portfolios of Workflows. The reduction method used here is based on the method in [1], 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 using a given performance measure (A3R, which combines accuracy and runtime) is straightforward. Identifying all algorithms with equivalent performance could be done with recourse to the Wilcoxon signed-ranks test, that exploits *fold* information of the cross-validation procedure. As we do not have this data, we had to use a substitute method instead. This method uses just the $N\%$ of top workflows as the most competitive algorithms for a given dataset. Here we use 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.

The aim of this phase is to eliminate all workflows which include rather infrequent OEM variants in this portfolio. If a particular OEM variant appears in less than P% of workflows, the corresponding workflows with this variant is marked for elimination. After processing all OEM variants, all corresponding workflows are dropped.

Algorithm/Workflow Recommendation Method Used. Here, we have chosen the method *average ranking (AR*)* [1] 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 [28], Auto-sklearn [10] 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 performance metadata. Each ranking is then followed to generate recommendations for the dataset left out. This enables to obtain its performance and to calculate how far it is from the best possible performance, i.e., calculate the *loss.* This is repeated as many times as there are datasets, following the leave-one-out (LOO) strategy. Sect. 5.3 shows the median loss obtained across all folds of LOO cycles.

Evaluation Strategy. The evaluation strategy adopted here is a leave-one-out (LOO) evaluation strategy. In each cycle, all datasets except one are used to identify the portfolios discussed above. The recommendations of the chosen algorithm selection method are used to calculate the loss on the dataset left out.

4 Experimental Setup

Our setup included 50 datasets from the *cc18* benchmark set of OpenML [31] (see Table 5 in the Appendix). We have not used datasets that were deemed to be too easy (the accuracy reported was higher than 95%) or those that had more than 50k instances.

In this study we have used 10 classifiers from the Weka toolkit that were used in one previous study [26] (see Table 1). Obviously, other classifiers could have been chosen (e.g., XGBoost, neural networks), but the choice made is useful for comparisons. All algorithms were used with default parameter settings. Apart from these, we have also used the *default classifier* that simply predicts the most frequent class for each dataset.

Table 1. Classifiers used in the experiments

Classifier	Description
IBk	5-Nearest Neighbors classifier [3]
J48	C4.5 Decision Tree classifier [22]
JRip	RIPPER propositional rule learner [8]
LMT	Classification trees with logistic regression at the leaves [18]
Logistic	Logistic regression model with a ridge estimator [19]
SimpleLogistic	Linear logistic regression model [27]
NaiveBayes	Naive Bayes using estimator classes [17]
PART	Generates rules based on partial Decision Tree leaves [11]
RandomForest	Random Forest classifier [6]
SMO	Sequential minimal optimization for SVM [21]

Twelve outlier detection and elimination methods (OEMs) have been used, some of which are general (see Table 2), others class-based [20] (see Table 3), representing a richer set than the one used in [25]. Each outlier elimination method (OEM) also has one hyperparameter indicating the percentage of top outliers to be eliminated (top 0.5, 1, 2, 3, 4 or 5%). All five values of this parameter were used in the experiments. Consequently, the total number of OEMs and its variants is 72, if we do not count the null method (12 OEMs, each with 6 hyperparameter settings).

The extended workflows have the format $OEM_{i,j}, CL_k$, where $OEM_{i,j}$ represents a particular outlier detection/elimination method i with a hyperparameter j. The total number of extended workflows was 720 (72 OEMs x 10 CLs). As

Table 2. General outlier detection methods used

OD	General outlier detection methods
LOF	Local Outlier Factor [7]: Compares the density of instances to its neighbors
NN	Nearest Neighbors [26]: Uses distances to the k nearest neighbors
IF	Isolation Forest [9]: Using forests, determines the outlyingness of instances based on their path lengths from the root to the isolation node
DS	Disjunct Size [26]: Outlyingness based on the size of leaf node of instances in the decision tree
TD	Tree Depth [26]: Outlyingness based on the depth of the leaf node using single decision tree
TDwP	Tree Depth with Pruning [26]: Same as TD but uses a pruned tree

each extended workflow was run on 50 datasets, the number of experiments was 36,000. To this, we need to also add the experiments with the initial workflows, which totaled 500 (10 CLs $\times$ 50 datasets). Each experiment was performed using 5-fold cross-validation.

5 Results

5.1 Can We Use OEMs Without Restrictions (RQ1)?

Our results have shown that, on average,[1] the workflows extended with outlier elimination do not exceed the initial counterpart. The only exception is Naive Bayes, whose performance, can, on average be improved by 0.316% by adding OEMs. So, the main conclusion from this experiment is that the OEMs should not be used blindly, without taking into account other aspects.

5.2 Determining Whether Some OEMs are Potentially Useful (RQ2)

Following the methodology defined in Sect. 3, we seek extended workflows of the type $OEM_{i,j}, CL_k$ whose performance exceeds the initial workflow CL_k on many datasets. In other words, our aim is to identify outlier methods $OEM_{i,j}$ that are *potentially useful* for a specific CL_k.

Some results of these experiments are shown in Table 4. This table shows some potentially useful workflows ($Classifier$, OEM) and the hyperparameter setting of the outlier method indicating how many elements should be left out (column *Out.*). Column *Init.* shows the average accuracy of the initial workflow (a particular classifier) on all datasets. The information in column *Extend.* is similar; it is relative to the workflow extended with the particular OEM. Column *Dif.* shows the difference between the two values. Positive values indicate

[1] The average is calculated by aggregating the accuracy across different OEMs and datasets.

Table 3. Class-based outlier detection methods used

OD	Class-based outlier detection methods
RF-OEX	Random Forest (RF) Outlier Detection and Explanation [20]: Uses *RF* to calculate the dissimilarity of instances to their own class, the similarity to other classes and general outlyingness
CODB	Class Outliers - Distance Based approach [13]: Uses nearest neighbors to calculate the dissimilarity of instances to their own neighborhood, the similarity to other classes and general outlyingness
KDN	K Disagreeing Neighbors [26]: Uses class labels of k nearest neighbors to calculate outlyingness
CLOF	Class-based Local Outlier Factor: Combines the dissimilarity to its own class, the similarity to other classes and general outlyingness: $LOF(sameclass) + 0.75 * LOF(otherclasses) + 0.25 * LOF(all)$
CL	Class Likelihood [26]: Calculates the probabilities of instances belonging to their own class based on Kernel Densities and the number of occurrences of features
CLD	Class Likelihood Difference [26]: CL, but with probabilities of belonging to different class also taken into account

Table 4. Some potentially useful workflows

Classifier	OEM	Out.	Init.	Extend.	Dif.	# Wins
			Acc.%	Acc.%	Acc.%	in 50
NaiveBayes	CODB	4	72.816	73.430	0.615	31
NaiveBayes	LOF	2	72.816	73.788	0.973	30
NaiveBayes	CODB	3	72.816	73.389	0.573	30
NaiveBayes	CODB	2	72.816	73.275	0.459	30
NaiveBayes	KDN	4	72.816	73.194	0.379	30
IBk	DS	5	79.490	79.659	0.169	30
LMT	ClassLikelihood	1	83.475	83.579	0.104	30
LMT	IsolationForest	0.5	83.475	83.554	0.079	30
SMO	RF-OEX	1	79.898	79.939	0.041	30
NaiveBayes	IsolationForest	2	72.816	73.793	0.978	29

that the particular OEM had a positive effect on performance. For instance, the use of outlier methods $CODB$ (with Out $= 4$), together with $NaiveBayes$ classifier, has led to an average increase of accuracy amounting to 0.615%. The improvements were observed on 31 out of 50 datasets (column $\#Wins$).

This analysis does not really show how to proceed. That is, if we selected a particular combination of $OEM_{i,j}, CL_k$ this would be a risky guess. As the experiments have shown, it is not guaranteed that this workflow would lead to a better performance. So, the only choice we have is to use the most promising alternatives and conduct tests on a validation set. This topic is discussed in the next section.

5.3 Constructing a Portfolio with the Most Useful Workflows with OEMs (RQ3)

In this subsection, we address the question of whether some of the extended workflows that include a particular *OEM* method are useful for algorithm recommendation.

As we explained in Sect. 3, we use the method based on average ranking AR* [1]. This method uses a ranked portfolio of workflows to generate the recommendations to the user. This makes it possible to compare the benefit of having workflows with OEMs in the portfolio. The results are shown in Fig. 1 showing the median curves that aggregate the data of different curves resulting from different cycles of LOO procedure.

All curves start with *default accuracy* for each dataset corresponding to the prediction of the most frequent class. The black loss curve (Baseline 10) includes the 10 base workflows and can be considered as the baseline.

The blue loss curve (Full Ranking 710) includes all 730 workflows. Ten of these are the initial workflows and 720 are the workflows that include OEMs (10 base workflows $\times$ 10 OEMs $\times$ 6 parameter settings = 720). The advantage of including OEMs is clearly visible. The corresponding curve reaches zero loss, while the black one that uses only classifiers does not. The difference when considering the median curves is rather significant - nearly 0.5%. The downside is that we need to spend more time (around 10^3 s) testing different alternatives before we encounter a good solution.

The red loss curve (Red Perc 1+1) shows the loss curve relative to the reduced portfolio that includes about 306 workflows on average, i.e., 58% reduction, which is quite significant. This portfolio was obtained by identifying the top performers for each dataset, then joining them without repetitions and constructing a single *A3R* ranking. The top performers include the top 1% of workflows based on A3R measure (it combines accuracy and time) and another top 1% of workflows based on accuracy only. So, this way, we identify 7+7 workflows per dataset. This alternative achieved a somewhat better loss as the total set represented by the blue curve. The advantage of this solution is that the portfolio includes fewer workflows.

The green loss curve (Red Perc 1+1 Subset) uses an even smaller portfolio when compared to the previous case (Red Perc 1+1). The portfolio Red Perc 1+1 is used as a starting point for this operation. The aim is to eliminate all workflows which include rather infrequent OEM variants in this portfolio. If a particular OEM variant appears in less than P% of workflows, the corresponding workflows with this variant is marked for elimination. After processing all OEM variants, all corresponding workflows are dropped. In this study, the threshold of P = 10% was used. The reduction obtained this way is significant, as it includes just 118 workflows (86% of reduction). Only 3 OEMs appeared in these workflows: RF-OEX, TDWithPrunning and DS.

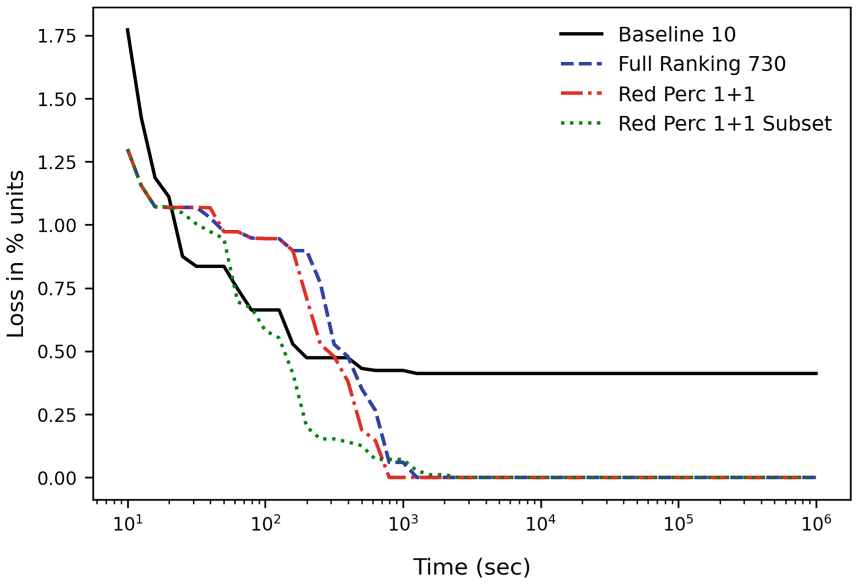

Fig. 1. Median loss curves of AR* relative to different portfolios of workflows (Color figure online)

6 Future Work and Conclusions

Future Work. It would be possible to examine how reliable are the OEMs identified in eliminating certain outliers. This could be done by injecting outliers in a controlled manner and then by examining whether these would be eliminated with the OEMs. Also, we could compare the effect of using OEMs with the noise filters discussed by Saez et al. [24] (see Sect. 2).

We note that outliers are often defined with respect to a particular distribution (see Sect. 3). In this work we have assumed that the distribution is fixed. It would be possible to extend the work presented here to be able to deal with data following a specific distribution. In case of skewed distributions, suitable transformations might help, e.g. Box-Cox, Yeo-Johnson or quantile-based transformations [15].

Although the reduction method used here was applied to a particular setting that includes OEMs, the method is quite general, as it can be applied in other settings. These could include algorithm selection, selection of suitable hyperparameter settings and inclusion of other preprocessing methods. The configuration space obtained with our specific workflow recommendation system can be implanted into other more complex systems. We are planning to conduct other studies in the future to demonstrate this.

Conclusions. Our aim was to examine the usefulness of outlier detection/elimination methods (OEMs) in classification. We have considered twelve different OEMs in conjunction with different classifiers and formulated several research questions. We have shown that it is not advisable to use OEMs in workflows without further restrictions. The performance of the workflows extended with OEMs does not usually exceed the performance of the initial counterparts. The workflows with Naive Bayes and OEMs represent an exception, although the gain obtained by including OEMs is not large.

This result lead us to investigate how we could identify the most useful workflows that include some OEMs only, and this way improve the performance. The methodology adopted used a simple algorithm/workflow recommendation system (AR*). We have investigated the effect of selecting different portfolios of workflows in this setting. Our aim was to try to reduce the initial portfolio by eliminating certain elements, without affecting the performance of the workflow recommendation system.

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%). Our most important result involves the three OEMs identified with our approach, which are RF-OEX, TDwP and DS. Besides, we showed that it is possible to eliminate 86% of the original workflows and still maintain the same loss.

Reducing the number of workflows can be regarded as a reduction of the given configuration space. This topic is relevant to other researcher in AutoML, as it can lead to substantial speed-ups of the search for effective solutions.

Acknowledgement. We thank Rita Ribeiro for her comments and suggestions, and Robert Kolcún, Ondrej Kurák and Adam Bajger for the implementation of the testing framework. This work was partially supported by the Faculty of Informatics, Masaryk University, Brno. This work was also partially financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020 and by the project Safe Cities - Inovação para Construir Cidades Seguras, with the reference POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement. We also thank the anonymous referees for their useful comments which enabled us to improve the paper.

Appendix

Table 5. Datasets used in the experiments (total 50)

Dataset	Inst.	Feat.	Clas.	Dataset	Inst.	Feat.	Clas.
dresses-sales	500	13	2	mfeat-fourier	2000	77	10
KC2	522	22	2	mfeat-karhunen	2000	65	10
climate-model-simulation-crashes	540	21	2	mfeat-morphological	2000	7	10
cylinder-bands	540	40	2	mfeat-pixel	2000	241	10
ilpd	583	11	2	mfeat-zernike	2000	48	10
balance-scale	625	5	3	KC1	2109	22	2
credit-rating	690	16	2	segment	2310	20	7
eucalyptus	736	20	5	ozone-level-8hr	2534	73	2
blood-transfusion-service	748	5	2	madelon	2600	501	2
pima-diabetes	768	9	2	dna	3186	181	3
analcatdata-dmft	797	5	6	splice	3190	61	3
vehicle	846	19	4	spambase	4601	58	2
tic-tac-toe	958	10	2	churn	5000	21	2
vowel	990	13	11	phoneme	5404	6	2
credit-g	1000	21	2	wall-robot-navigation	5456	25	4
qsar-biodeg	1055	42	2	texture	5500	41	11
cnae9	1080	857	9	optdigits	5620	65	10
PC1	1109	22	2	first-order-theorem	6118	52	6
pc4	1458	38	2	satimage	6430	37	6
cmc	1473	10	3	data.va3.gesture	9873	33	5
pc3	1563	38	2	JM1	10885	22	2
semeion	1593	257	10	letter	20000	17	26
car	1728	7	4	doushouqi-raw-egtb-2-pieces	44819	7	3
spf3	1941	28	7	bank-marketing-full	45211	17	2
mfeat-factors	2000	217	10	electricity	45312	9	2

References

1. Abdulrahman, S.M., Brazdil, P., van Rijn, J.N., Vanschoren, J.: Speeding up algorithm selection using average ranking and active testing by introducing runtime. Mach. Learn. **107**(1), 79–108 (2017). https://doi.org/10.1007/s10994-017-5687-8
2. Abdulrahman, S.M., Brazdil, P., Zainon, W.M.N.W., Adamu, A.: Simplifying the algorithm selection using reduction of rankings of classification algorithms. In: ICSCA 2019 Proceedings of the 2019 8th International Conference on Software and Computer Applications, pp. 140–148. ACM, New York (2019)
3. Aha, D., Kibler, D.: Instance-based learning algorithms. Mach. Learn. **6**, 37–66 (1991)
4. Barnett, V., Lewis, T.: Outliers in Statistical Data. Wiley, Hoboken (1978)
5. Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J.: Metalearning: Applications to Automated Machine Learning and Data Mining, 2nd edn. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-030-67024-5
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

7. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: ACM SIGMOD Record, vol. 29, pp. 93–104. ACM (2000)
8. Cohen, W.W.: Fast effective rule induction. In: Twelfth International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann (1995)
9. Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proc. Vol. **46**(20), 12–17 (2013)
10. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 113–134. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_6
11. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Shavlik, J. (ed.) Fifteenth International Conference on Machine Learning, pp. 144–151. Morgan Kaufmann (1998)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The Weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
13. Hewahi, N.M., Saad, M.K.: Class outliers mining: distance-based approach. Int. J. Intell. Syst. Technol. **2**, 5 (2007)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. LION **5**, 507–523 (2011)
15. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice, 3rd edn. OTexts (2021)
16. John, G.H.: Robust decision trees: removing outliers from databases. In: Knowledge Discovery and Data Mining, pp. 174–179. AAAI Press (1995)
17. John, G.H., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338–345. Morgan Kaufmann, San Mateo (1995)
18. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Mach. Learn. **59**(1), 161–205 (2005)
19. le Cessie, S., van Houwelingen, J.C.: Ridge estimators in logistic regression. Appl. Stat. **41**(1), 191–201 (1992)
20. Nezvalová, L., Popelínský, L., Torgo, L., Vaculík, K.: Class-based outlier detection: staying zombies or awaiting for resurrection? In: Fromont, E., De Bie, T., van Leeuwen, M. (eds.) IDA 2015. LNCS, vol. 9385, pp. 193–204. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24465-5_17
21. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schoelkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)
22. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
23. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976)
24. Sáez, J.A., Corchado, E.: ANCES: a novel method to repair attribute noise in classification problems. Pattern Recogn. **121**, 108–198 (2022)
25. Smith, M.R., Martinez, T.: The robustness of majority voting compared to filtering misclassified instances in supervised classification tasks. Artif. Intell. Rev. **49**(1), 105–130 (2016). https://doi.org/10.1007/s10462-016-9518-2
26. Smith, M.R., Martinez, T., Giraud-Carrier, C.: An instance level analysis of data complexity. Mach. Learn. **95**(2), 225–256 (2013). https://doi.org/10.1007/s10994-013-5422-z

27. Sumner, M., Frank, E., Hall, M.: Speeding up logistic model tree induction. In: Jorge, A.M., Torgo, L., Brazdil, P., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 675–683. Springer, Heidelberg (2005). https://doi.org/10.1007/11564126_72
28. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855. ACM (2013)
29. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **6**, 448–452 (1976)
30. van Rijn, J.N., Hutter, F.: Hyperparameter importance across datasets. In: KDD 2018: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM (2018)
31. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49–60 (2013)
32. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artif. Intell. Rev. **18**(2), 77–95 (2002)
33. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. **38**(3), 257–286 (2000)
34. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Evaluating component solver contributions to portfolio-based algorithm selectors. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 228–241. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31612-8_18
35. Zhu, X., Wu, X.: Class noise vs. attribute noise: a quantitative study. Artif. Intell. Rev. **22**, 177–210 (2004)