Neurocomputing 486 (2022) 77–92

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Supervised outlier detection for classification and regression

Ángela Fernández [a,*], Juan Bella [a], José R. Dorronsoro [a,b]

[a] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, 28049, Spain
[b] Instituto de Ingeniería del Conocimiento, 28049 Madrid, Spain

ARTICLE INFO

Article history:
Received 4 April 2021
Revised 11 January 2022
Accepted 19 February 2022
Available online 25 February 2022

Keywords:
Outlier detection
Optimal hyperparameter selection
Supervised learning
Classification
Regression
Histogram based outlier detection
Minimum covariance determinant
Local outlier factor
Isolation forests

ABSTRACT

Outlier detection, i.e., the task of detecting points that are markedly different from the data sample, is an important challenge in machine learning. When a model is built, these special points can skew the model training and result in less accurate predictions. Due to this fact, it is important to identify and remove them before building any supervised model and this is often the first step when dealing with a machine learning problem. Nowadays, there exists a very large number of outlier detector algorithms that provide good results, but their main drawbacks are their unsupervised nature together with the hyperparameters that must be properly set for obtaining good performance. In this work, a new supervised outlier estimator is proposed. This is done by pipelining an outlier detector with a following a supervised model, in such a way that the targets of the later supervise how all the hyperparameters involved in the outlier detector are optimally selected. This pipeline-based approach makes it very easy to combine different outlier detectors with different classifiers and regressors. In the experiments done, nine relevant outlier detectors have been combined with three regressors over eight regression problems as well as with two classifiers over another eight binary and multi-class classification problems. The usefulness of the proposal as an objective and automatic way to optimally determine detector hyperparameters has been proven and the effectiveness of the nine outlier detectors has also been analyzed and compared.

1. Introduction

A frequent challenge in Machine Learning (ML) problems is the presence of individual patterns that substantially differ from the majority of sample instances. Such anomalous patterns are often termed as outliers which, according to Hawkins [1] and as quoted in [2], can be defined as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism".

Sometimes the term anomaly also arises in this context with more or less the same meaning, the main difference being the problem to be addressed [3]. In general, the goal in Outlier Detection (OD) is to detect and separate the outliers in a sample from the normal data; in other words, the sample is considered as "contaminated" and the OD method has to "clean" it before its use as a training sample. On the other hand, the goal in anomaly detection [4] is to characterize an already clean sample and use it to detect those new patterns that do not follow it. The situation in this work will be the first one, in which the main assumption is that the normal patterns, i.e., the *inliers*, are generated according to a certain procedure while the *outliers* are not. The natural goal under this assumption would be to estimate this "normal" generative model. However, this may be in general very difficult and the alternative is to build models that either assign an inlier/outlier label to the patterns or, more generally, compute an *outlierness score* that, if needed, can further be binarized through a threshold.

In principle, OD is an unsupervised problem, as usually there is no previous knowledge of which patterns are the outliers. On the other hand, OD models rely on a number of hyperparameters whose values are in general crucial for the model performance [5]. For instance, and as mentioned before, OD algorithms build a scoring function to quantify how far a given pattern is from the sample general structure [6]. Once available, these scores allow us to rank the sample's patterns and to establish a cut-off point as the boundary between normal patterns and outliers [7]. For this purpose, OD methods usually define a contamination parameter which is set as the fraction of total sample size to be considered as outliers.

This variable is only the first of a series of hyperparameters that have to be optimally tuned for a best performance. Unfortunately, the unsupervised nature of OD models implies the absence of a ground truth to be used for separating outliers from inliers and,

20pt Corresponding author at: C/Francisco Tomás y Valiente, 11, E.P.S., Edificio B, 4 planta, UAM-Cantoblanco, 28049 Madrid, Spain.

E-mail addresses: a.fernandez@uam.es (Á. Fernández), juan.bellas@estudiante.uam.es (J. Bella), jose.dorronsoro@uam.es (J.R. Dorronsoro).

https://doi.org/10.1016/j.neucom.2022.02.047
0925-2312/

as a consequence, it is quite difficult to perform an optimal hyperparameter estimation. In some application-specific problems a subset of such anomalous patterns may be known and then the OD problem can be turned into a supervised one, often to be dealt as a classification problem; this is usually the situation referred as supervised OD. But even in this case, OD is often the first step within a data modeling process which has a supervised classifier or regressor as its final step.

In such situations it is often the case that optimal hyperparameters for the OD model and the classifier or regressor are obtained successively but independently; that is, the OD hyperparameters are obtained first and, once the training sample is cleaned, the classifier or regressor hyperparameters are then estimated. But given that the OD and the supervised model have to work together, a natural idea is to take advantage of this joint working to exploit the targets, not only to get the supervised model hyperparameters, but also those of the OD model as well.

We will follow this approach here by considering the composition of an OD algorithm $D(x;\alpha)$ that acts on a pattern x and has an hyperparameter set α with a supervised classifier or regressor $M(x;\beta)$, hyperparametrized now by the set β. More precisely, the D operator decides whether or not a pattern x is an outlier. If it is not, $D(x,\alpha) = x$ and M acts on it. If, on the other hand, D detects x as an outlier, M is not applied to it.

This can be understood as a pipeline model acting on an $N \times d$ sample matrix X with the transpose of the p-th pattern $x^p, 1 \leqslant p \leqslant N$, in its p-th row, i.e., $X_p = (x^p)^t$. With a slight abuse of language, $X^{cl} = D(X;\alpha)$ represents the $N_{cl} \times d$ matrix made of the N_{cl} patterns that D has deemed not to be outliers. Then, the pipelined model is defined as: $P(X;\alpha,\beta) = M(D(X;\alpha);\beta) = M\left(X^{cl};\beta\right)$, which outputs a prediction vector or matrix $\widehat{Y}^{cl}$ to be compared with the target Y^{cl} associated with the "clean" vectors selected by D. The optimal hyperparameters (α,β) are selected via cross validation on the Cartesian product $R_\alpha \times R_\beta$ of appropriately chosen α,β ranges R_α, R_β (more details are given in Section 3). Notice that, in the preceding, D can be in principle any OD method and M any classification and regression algorithm. This lends to the proposal a high degree of flexibility to combine the D and M algorithms best suited to the problem at hand. We will illustrate this in the experiments, where nine different OD algorithms will be considered jointly with two classification and three regression ones.

It must be stressed that, while the proposal places the OD model D under a supervised setting, it is very different from other supervised approaches that have been followed in OD such as, for instance, those in [8,9] or Chapter 7 of [2]. In fact, here no prior information is assumed to be known about anomalies in the training data, i.e., the independent variables where OD will take place, and the only supervision used comes from the classification or regression target vector Y, i.e., the dependent variable. In other words, and with respect to the training sample matrix X, we are under an unsupervised OD scenario. In contrast, in standard supervised OD, concrete examples of normal and anomalous patterns are known, which enables to deal with the OD problem as a classification one (often with highly unbalanced classes, as anomalies will be much less frequent than normal patterns).

In summary, and besides a short overview of relevant OD algorithms from an application point of view which may have an interest on its own, the main contributions of this article are:

- The proposal of combining OD algorithms with supervised regression and classification ones in such a way that optimal OD hyperparameters can be chosen jointly with the supervised ones.
- Extensive experiments over eight classification and regression datasets on the performance of pipelines with nine algorithms on its OD component and five algorithms on its modeling component.
- The numerical justification of the improvement of base classifiers and regressors when they are pipelined with OD models and an experimental framework where the performances of conceptually quite different OD models can be objectively compared.

The rest of the paper is organized as follows. In Section 2 we will first briefly review the literature on OD models, after which the OD models used in this work will be described in more detail in a series of subsections. Section 3 contains our proposal for supervised outlier detection. In Section 4 our experimental methodology is described, as well as the datasets used, and the results of the regression and classification experiments are presented, together with some considerations on execution times. Finally, in Section 5 the main results will be discussed, some conclusions offered and some pointers to further work given.

2. Outlier Detection

As already said, Outlier Detector (OD) is a huge research area and it is extremely difficult to have all methods compared in a single experimental setting. Because of this, in this work nine different outlier detection methods with a prominent role in the literature have been selected and compared, using the implementations available in either *scikit-learn* [10] or *PyOD* [11], two well-known Python libraries. In this section a brief overview of the literature on OD methods will be given first, with an emphasis in the concrete methods we use but mentioning also a series of papers dealing with statistical aspects of OD as well as the latest contributions on the application of Deep Neural Networks (DNN), since many are more or less inspired on the Principal Component Analysis approach to OD, one of the methods we consider. This is the content of SubSection 2.1 after which a more detailed individual presentation of the OD methods used here will be given. All of them share a common parameter: the contamination percentage, i.e., the proportion of outliers we believe there are in the sample. In general, this parameter is used to define the threshold on the decision function. The other hyperparameters which will be optimized will also be discussed, leaving the remaining ones at their library defaults.

2.1. Literature Review

From a general point of view, probably the most comprehensive reference is the book by C. Aggarwal [2] which addresses not only the most representative OD algorithms but also their extensions to ensemble settings [12], approaches to OD based on supervised ideas [8,9] (although quite different from the ones here) and also their application to problems involving categorical [13,14], text [15] or spatial data [16], time series [17] or graphs [18]. Extensive bibliography reviews can be found at the end of the corresponding chapters in [2]. Other interesting general reviews are [19,5,20].

Going into concrete approaches, we shall consider here nine representative OD models, namely, Minimum Covariance Determinant estimator [21,22], Local Outlier Factor [23], Connectivity-based Outlier Factor [24], k-Nearest Neighbors outlier estimator [25,26], Isolation Forests [27,28], One-class Support Vector Machine [29], Principal Components Analysis (PCA) OD [30,31], Subspace Outlier Detection [32] and Histogram-based Outlier Detection [33]. More details on them are given in the next subsections but, in any case, they are part of the vast literature on OD models.

Looking at this variety from an application point of view, a very good reference is the *PyOD* library [11], from which we have used the implementations of all the above methods except Minimum Covariance Determinant and Isolation Forest, for which we use their *scikit-learn* [10] implementations. Besides them, there must be mentioned methods such as Angle [34] or Copula [35] based algorithms, or other methods based on classification or regression trees such as [36] or on ensemble based approaches, such as feature bagging OD [37].

Recently many approaches to OD based on DNN have been proposed; a recent survey on this topic is [38]. Many of these methods take as their basis deep autoencoders that extend the PCA based OD algorithm. Examples of this are [39–41] or [42]. In the same line, it can be also mentioned the proposals based on generative or adversarial neural autoencoders in [43,44] and the survey in [45]. Neural networks have also been considered for one class based OD [46]; other interesting references on this line are [47,48].

Finally, relevant contributions that deal with the more statistical aspects of OD methods are [6], which discusses how to unify the scores usually returned by OD methods, [49], with a statistical analysis of nearest neighbor based OD algorithms, or [50] which examines from a general point of view the causes and sources of the presence of outliers and how to mitigate them.

We turn next to the details of the OD methods that will be used here.

2.2. Minimum Covariance Determinant

The Minimum Covariance Determinant (MCD) estimator [21,22] looks for the h samples with a minimum determinant of their covariance matrix, and then it computes the Mahalanobis distance for determining the outlier points. It is based on the Minimum Volume Ellipsoid (MVE) estimator [21], that looks for the ellipsoid of minimum volume that contains h of the n observations, where $n/2 \leqslant h < n$.

The MCD estimator has been designed to be applied on Gaussian-distributed data, but could still be relevant on data drawn from a unimodal, symmetric distribution. In general it is hard to compute, but luckily there exist a FAST-MCD version [21], and this is the algorithm that we will use, as implemented in *scikit-learn*. When applying it, we have to estimate the value of the support fraction ρ (added to the contamination percentage). This hyperparameter represents the proportion of points to be included in the support (the ellipsoid) of the raw MCD estimation.

2.3. Local Outlier Factor

The Local Outlier Factor (LOF, [23]) estimator measures the local density deviation of a given sample with respect to its neighbors. We consider then an outlier as a sample point that has a substantially lower density than their neighbors. A formal definition of the so-called LOF_k measure is

$$LOF_k(x_i) = \frac{1}{k}\left(\sum_{x_l \in N_k(x_i)} \mathrm{k}-\mathrm{dist}(x_l)\right)\left(\sum_{x_j \in N_k(x_i)} \frac{1}{\sum\limits_{x_m \in N_k(x_j)} \mathrm{k}-\mathrm{dist}(x_m)}\right), \tag{1}$$

where $\mathrm{k}-\mathrm{dist}(x_i)$ can be approximately defined as the distance of x_i to its farthest point in the set $N_k(x_i)$ of its k neighbors.

By observing (1), we can conclude that when a point is an inlier, it should be "well-located" inside its neighborhood and hence, its LOF value will be near to 1. On the other hand, an outlier will be located far away from its neighbors resulting in $LOF(x_i) \gg 1$. Therefore, the larger the $LOF_k(x_i)$, the more anomalous x_i can be considered. Here we need to choose the number of neighbors k to be considered, besides, as usual, the contamination hyperparameter.

2.4. Connectivity-Based Outlier Factor

The Connectivity-based Outlier Factor (COF, [24]) method is a refinement of the above LOF algorithm, in the sense that low-density points are not always considered as isolated points (i.e. outliers). The COF algorithm is focused on the connectivity pattern of each point with respect to its neighbors, defining a score as the following ratio: $COF_k(p) = \frac{|N_k(p)|\mathrm{ac-dist}_{N_k(p)}(p)}{\sum_{o \in N_k(p)} \mathrm{ac-dist}_{N_k(o)}(o)}$,where $\mathrm{ac}-\mathrm{dist}_{N_k(p)}(p) = \sum_{i=1}^{k-1} \frac{2(k-i)}{k(k-1)} dist(e_i)$ is the average chaining distance from p to its nearest k neighbors in $N_k(p)$ and e_i are the edges involved. In this case we used the implementation of the *PyOD* library, looking for the optimal number of neighbors k and the contamination percentage.

2.5. k-Nearest Neighbors

The k-Nearest Neighbors OD [25,26] is a distance-based method which computes a score based on the distances to the k-th nearest neighbors per point, defining as outliers the points with the highest scores. In this sense the method is also measuring the density of the sample. The outlying score can be computed by calculating the average of all the k neighbors, its median or by just taking into account the distance to the k-th neighbor. We have applied the *PyOD* version of this method, hyperparametrizing the number k of nearest neighbors and the contamination.

2.6. Isolation forests

The previous methods try to identify what the "normality" is and then classify as anomalies the patterns that do not follow this property [27]. But these models can be inefficient for detecting anomalies, and their complexity grows with the size and dimension of the dataset.

To prevent these drawbacks, Isolation Forests (iForests, [28]) try to explicitly isolate the anomalies, instead of focusing on defining normality. The iForest algorithm builds a set of iTrees and defines as outliers those points that, in mean, have lower depth. Moreover, it is not a high cost algorithm in terms of computational complexity, as it is not necessary to build the trees completely, it does not use distance metrics and it does not need a large amount of memory.

We have used the implementation of this algorithm given by *scikit-learn*, where the hyperparameters to be estimated are the number of iTrees and the maximum number of features per tree node, plus the contamination percentage.

2.7. One-class SVM

One-class Support Vector Machine (One-class SVM, [29]) is a kernel-based anomaly detector that ultimately tries to estimate the density function of the data and then to define a binary function f which determines whether or not a point belongs to the region where the sample lies. In practice, and as for standard SVMs, we have to solve a dual optimization problem formally defined as

$$\min_{\alpha \in \mathbb{R}^N} \left\{\frac{1}{2}\alpha^{\intercal} K \alpha\right\} \text{s.t.} \begin{cases} \alpha^{\intercal}\mathbf{1} = 1, \\ \mathbf{0} \leqslant \alpha \leqslant \frac{1}{\nu N}, \end{cases}$$

where:

- K is the kernel function. In general we will use the Gaussian kernel defined as $k(x, y) = e^{-\gamma\|x-y\|^2}$, where γ is the kernel scale. A good approximation to its optimal value can be obtained by $1/d$, with d the number of features; we will use this value after scaling pattern features to the $[0, 1]$ range.
- $\nu \in (0, 1]$ measures the anomaly detection sensitivity. It is an upper-bound of the fraction of errors allowed, and a lower-bound of the number of support vectors. A large ν implies a small support for the sample data (so almost every point will be considered an anomaly) and a small ν provides a large support and some anomalies can be missed, so this is an important hyperparameter to be optimized.

In this case, we have used for the experiments the implementation of this method provided by *PyOD*, where the model is used as an outlier detector and allows to tune the contamination percentage as well as the ν parameter just explained.

2.8. Principal Components Analysis OD

Principal Components Analysis (PCA) can be also used as an outlier detector method [30,31]. In this case, the points considered as outliers are the ones that have a large variance in comparison with the "normal" patterns. This variance distance is even more obvious over the projected hyperplane of PCA. Because of this, the outlying score is defined in this case as the sum of Euclidean distances from each sample to the smaller-dimensionality projected hyperplane (constructed from the main eigenvectors of the correlation matrix).

We used the method provided by *PyOD*, hyperparametrizing the contamination percentage and, indirectly, the number of components to be used through the variance percentage they explain.

2.9. Subspace Outlier Detection

The Subspace Outlier Detection (SOD, [32]) schema aims to detect outliers in varying subspaces of a high dimensional feature space. The general idea of this method is to check how well a point fits with respect to the subspace spanned by a set of reference points selected from its neighbors. An outlier is defined as a point that deviates far enough from its reference hyperplane. The score in this case at a point p is defined in terms of the reference set as: $SOD_{R(p)}(p) = dist\left(\frac{p, \mathscr{H}(R(p))}{\|v^{R(p)}\|_1}\right)$,where $\mathscr{H}(R(p))$ is the hyperplane spanned by its reference set $R(p)$ and $v^{R(p)}$ are the subspace defining vectors of the reference set which specify the most relevant attributes, i.e., the ones with lower variance.

We have used the implementation of this function offered by *PyOD* where, besides the contamination percentage, we hyperparametrize the number of neighbors k, leaving the size of the reference set to the minimum of 10 and $k-1$, as it must be smaller than k.

2.10. Histogram-based Outlier Detection

Histogram-Based Outlier Detection (HBOS, [33]) is a combination of univariate methods, namely the histograms of each feature, assuming feature independence. Its outlier score is computed as: $HBOS(p) = \sum_{i=1}^{d} \log\left(\frac{1}{\text{hist}_i(p)}\right)$.It has the advantage to be very fast in comparison with other outlier detectors as its score computation has a linear cost. We have used in this case the implementation given in the *PyOD* library, where apart from the contamination percentage, we optimize a regularization parameter, leaving other parameters, such as the number of bins of the histogram, to the library's defaults.

3. Supervised Outlier Detection

In this scenario, a classification or regression problem is given with a sample matrix X and a target vector Y, where X is possibly contaminated and the target is to clean it before building the final classification or regression model. In order to have a more concise exposition, such model will be referenced simply as a *supervised model* to avoid lengthy repetitions of classification or regression; moreover, and in a slight abuse of language, X and Y will denote not only the input matrix and target vector, respectively, but also their corresponding samples. The new proposal is to solve the underlying problem via a pipeline made up of an OD algorithm $D(x;\alpha)$ and a supervised model $M(x;\beta)$. In this way, D will be applied to X to get a clean version X^{cl} as well as its corresponding target vector Y^{cl}.

In principle it is conceivable that the OD algorithm could be applied to the joint $\{X_{tr}, Y_{tr}\}$ training sample, but this would lead to a conflict when the OD models were to be applied to a new test sample X_{ts} for which their targets Y_{ts} would not be known at the prediction moment. In other words, if we train and build an OD algorithm D on both X_{tr}, Y_{tr}, it has to be tweaked somehow so that it can be applied during testing when only the test sample X_{ts} is known but Y_{ts} is not. We can do this for instance for k-NN based OD methods, as we only need the number k of neighbors to be used: while it is estimated on the train pair X_{tr}, Y_{tr}, k-NN could be also applied in principle to just X_{ts}. However, such a tweaking is impossible for other methods. A clear example is PCA-based OD. A PCA projection matrix computed on the $d+1$ dimensional X_{tr}, Y_{tr} sample, would have m rows, with m the number of principal components retained, and $d+1$ columns. But it is obvious that this $m \times (d+1)$ projection matrix cannot be applied on the test sample X_{ts}, as it only has d dimensions, since we are missing the extra dimension that would come from the unknown test targets.

This is one clear reason to build the OD methods only on the X_{tr} data matrix and not on the training pair X_{tr}, Y_{tr}, but some other important ones can be added. For instance, while target anomalies may be natural in regression problems as due for instance to incorrect measurements, in classification ones they just would correspond to mislabeled patterns. Also, classification targets are essentially categorical and, hence, some codification is needed before the OD method is to be applied, but which may result in hiding or masking possible anomalies.

Because of this, in this work OD we be will only applied to the train $X = X_{tr}$ and, eventually, to the test X_{ts} samples. Notice that although no OD algorithm is directly applied to Y_{tr} to get Y^{cl}_{tr}, Y_{tr} is indirectly cleaned just by the selection of the final input-target pairs (x, y) with $x \in X^{cl}_{tr}$. It should be also pointed out that there are regression algorithms such as the Huber, RANSAC or Theil-Senn regressors, that are designed to be robust with respect to target outliers. In fact, we have applied one of them, Huber regression, in our experiments.

3.1. OD pipelined model

Recall that in this work it is proposed to define a pipelined model in the form $P(X;\alpha,\beta) = M(D(X;\alpha);\beta) = M\left(X^{cl};\beta\right)$ acting on a sample matrix X and where X^{cl} is the subsample matrix made up of the "clean" vectors selected by D. 7 Assuming to have a training pair $\{X_{tr}, Y_{tr}\}$ and another test pair $\{X_{ts}, Y_{ts}\}$, and having selected optimal hyperparameters α^*, β^*, this pipeline can be exploited under two scenarios. In the first one M is trained on the clean train sample $X^{cl}_{tr} = D(X_{tr};\alpha^*)$ but it is applied to the entire test sample X_{ts}, which it is assumed to be clean, obtaining $\widehat{Y}_{ts} = M(X_{ts};\beta^*)$. In the second scenario, X_{ts} could not be clean, and hence M is applied to its clean version $X^{cl}_{ts} = D(X_{ts};\beta^*)$ to obtain $\widehat{Y}^{cl}_{ts} = M(D(X_{ts};\alpha^*);\beta^*)$.

Both scenarios are presented in Algorithm 1, where a flag fl_{cl} is used to select which one is to be applied. Notice that when fl_{cl} is 1, the OD algorithm D is applied to both the train X_{tr} and test X_{ts} sample matrices, while it is only applied to X_{tr} when fl_{cl} is 0.

Algorithm 1: $\texttt{PipelineOD}(X_{tr}, Y_{tr}, X_{ts}, D, M, \alpha^*, \beta^*, fl_{cl})$

1. $D(\cdot;\alpha^*).\texttt{fit}(X_{tr})$
   #clean X_{tr} and select train targets for M
2. $X^{cl}_{tr} = D(\cdot;\alpha^*).\texttt{predict}(X_{tr})$
3. $Y^{cl}_{tr} = \texttt{select}\left(Y_{tr}, X^{cl}_{tr}\right)$
   #fit M on a clean train sample
4. $M(\cdot;\beta^*).\texttt{fit}\left(X^{cl}_{tr}, Y^{cl}_{tr}\right)$
5. **if** $fl_{cl} == 0$ **then**
   #predict on the test sample without cleaning it
6. $\widehat{Y}_{ts} = M(\cdot;\beta^*).\texttt{predict}(X_{ts})$
7. return $\widehat{Y}_{ts}$
8. **else if** $fl_{cl} == 1$ **then**
   #clean the test sample and ...
9. $X^{cl}_{ts} = D(\cdot;\alpha^*).\texttt{predict}(X_{ts})$
   #predict on the clean test sample
10. $\widehat{Y}^{cl}_{ts} = M(\cdot;\beta^*).\texttt{predict}\left(X^{cl}_{ts}\right)$
11. return $\widehat{Y}^{cl}_{ts}$

It remains to specify how to choose the optimal α^*, β^* hyperparameters. In this work it is proposed to do it by K-fold cross validation over the product $R_\alpha \times R_\beta$, with R_α, R_β the α and β hyperparameter ranges, and it will depend on the scenario to be used, with D being applied only on a training fold but not on a validation (or test) one when fl_{cl} is 0, but to both when fl_{cl} is 1. This methodology is specified in Algorithm 2 which, in turn, relies on Algorithm 1. Notice again that when fl_{cl} is 1, D will be applied to both the training X'_k sample matrices when Algorithm 1 is called, and also to the validation matrices X_k.

Algorithm 2: $\texttt{CV}(X_{tr\_val}, Y_{tr\_val}, D, M, R_\alpha, R_\beta, fl_{cl})$

1: Split (X_{tr_val}, Y_{tr_val}) into K folds (X_k, Y_k)
2: Select a classification or regression loss $\ell(\cdot,\cdot)$
3: **for** $(\alpha,\beta) \in R_\alpha \times R_\beta$ **do**
4: **for** $k = 1, \ldots, K$ **do**
   #compute score $sc_k(\alpha,\beta)$ on the k-th CV fold
5: $X'_k = \cup_{j\neq k} X_j$, $Y'_k = \cup_{j\neq k} Y_j$
6: **if** $fl_{cl} == 0$ **then**
7: $\widehat{Y}_k = \texttt{PipelineOD}(X'_k, Y'_k, X_k, D, M, \alpha, \beta, fl_{cl} = 0)$
8: $sc_k(\alpha,\beta) = \ell\left(Y_k, \widehat{Y}_k\right)$
9: **elseif** $fl_{cl} == 1$ **then**
10: $\widehat{Y}^{cl}_k = \texttt{PipelineOD}(X'_k, Y'_k, X_k, D, M, \alpha, \beta, fl_{cl} = 1)$
   #clean X_k and select test targets for M
11: $X^{cl}_k = D(\cdot;\alpha).\texttt{predict}(X_k)$
12: $Y^{cl}_k = \texttt{select}\left(Y_k, X^{cl}_k\right)$
13: $sc_k(\alpha,\beta) = \ell\left(Y^{cl}_k, \widehat{Y}^{cl}_k\right)$
   #compute the mean score for each $(\alpha,\beta) \in R_\alpha \times R_\beta$
14: $\overline{sc}(\alpha,\beta) = \{sc_k(\alpha,\beta)\}.\texttt{mean}_k$
#compute optimal (α^*,β^*) with the minimum score
15: $(\alpha^*,\beta^*) = \{\overline{sc}(\alpha,\beta)\}.\texttt{arg\_min}_{\alpha,\beta}$
16: return (α^*,β^*)

3.2. Computational cost analysis

The previous considerations are relevant when the complexity of D and M are taken into account. This may be quite involved as we will work with nine OD algorithms D and five supervised models M but, from a general point of view, let $c^f_D(N), c^p_D(N)$ and $c^f_M(N), c^p_M(N)$ be the fit and predict costs, respectively, of D and M when applied to sample matrices with N patterns. In general, one could assume the predict cost to be much smaller than the fit costs; this is the case of the Ridge, Huber, Logistic regression and MLP models used, and their prediction costs $c^p_M(N)$ will not be considered. However, this is not the case for several OD models such as One-Class SVM and KNN and their costs $c^p_D(N)$ will be retained in what follows.

Now, denoting by $c_{D,M}(N_{tr}, N_{ts}, fl_{cl})$ the cost of applying Algorithm 1 over training and test samples with sizes N_{tr}, N_{ts}, respectively, and using the flag fl_{cl}, we have

$$\begin{aligned} c_{D,M}(N_{tr}, N_{ts}, fl_{cl}) &= c^f_D(N_{tr}) + c^p_D(N_{tr}) + c^f_M(N_{tr}) \text{ if } fl_{cl} = 0; \\ &= c^f_D(N_{tr}) + c^p_D(N_{tr}) + c^f_M(N_{tr}) + c^p_D(N_{ts}) \text{ if } fl_{cl} = 1. \end{aligned}$$

Taking this into account, and denoting now by $c^{CV}_{D,M}(N, R_\alpha, R_\beta, K, fl_{cl})$ the cost of applying Algorithm 2 over a size N sample, the ranges R_α, R_β with K folds and a flag fl_{cl}, we have (with a slight abuse of language)

$$\begin{aligned} c^{CV}_{D,M}(N, R_\alpha, R_\beta, K, fl_{cl}) = &\ |R_\alpha| \times |R_\beta| \times K \times c_{D,M}\left(\tfrac{K-1}{K}N\right) \text{ if } fl_{cl} = 0, \\ &\ |R_\alpha| \times |R_\beta| \times K \times \left(c_{D,M}\left(\tfrac{K-1}{K}N\right) + c^p_D\left(\tfrac{N}{K}\right)\right) \text{ if } fl_{cl} = 1, \end{aligned}$$

where $|R_\alpha|, |R_\beta|$ denote the number of α, β hyperparameters to be considered. As it may be expected, when OD algorithms D with substantial c^f_D or c^p_D costs are involved, CV hyperparametrization may be very costly even for moderately sized datasets. This fact will be briefly commented in the next Section on the pipelined execution times of the OD algorithms used. However, it must be stressed that actual execution times depend not only on the theoretical costs of the D and M algorithms but also on their implementations. This is particularly the case when Python is involved, as execution times of methods directly implemented in Python will be much higher than those that use Python just as a wrapper around more efficient implementations (as is, for instance, the case of the One-class SVM, that relies on its C++ implementation in the LIBSVM library).

4. Numerical Experiments

In this section our experimental methodology will be described first, the datasets used will be then discussed in some detail, after which the regression and classification results will be presented and the section will finish with a brief execution time comparison for the OD models.

As mentioned before, the OD models applied in this work are `mcd`, `if`, `knn`, `lof`, `ocs`, `pca`, `cof`, `hbos` and `sod`, using their *scikit-learn* implementation in the case of `mcd` and `if` and their *PyOD* versions for the others. Recall that they have been briefly described in Section 2. Notice that by setting the contamination parameter value to be 0, no data filtering is performed, and our approach just reduces to the straight application of the linear and nonlinear classifiers and regressors; this case is identified as the `base` model. The regression supervised models will be Ridge and Huber regression as linear regression models, and a non linear multilayer perceptron regressor with ReLU activations and a single hidden layer with 20 units. For classification logistic regression will be used, in its standard formulation for two class problems and in its multinomial version for the multiclass problems; a non linear multilayer classifier again with ReLU activations and a single hidden layer with 20 units will be also used. This choice is partly made to alleviate computational costs but also, given that the selected datasets have moderate sample sizes and number of dimensions, to yield good classification and regression results. The implementations in *scikit-learn* have been selected for all the classifiers and regressors.

4.1. Experimental Methodology

This section describes the experimental analysis done for testing the proposed supervised hyperparametrization of OD models. We basically follow the procedure in Algorithm2; in the notation of that algorithm, the OD models listed above are taken as the D models. In the same notation, the supervised ones just mentioned will be used as the M models. The samples are feature-wise normalized to a $\mu = 0, \sigma = 1$ distribution for all the experiments except the ones where one-class SVM is involved, for which features are scaled to the $[0, 1]$ range. We recall that in the experiments presented we will only filter the data matrix X. On the other hand, results will be presented under two different testing scenarios. In the first one, the training matrix X_{tr} is cleaned but the OD component D is not applied to the validation or test sets, but only the supervised model M; i.e., the test set X_{ts} is assumed to be clean. In the second one, the goal is to clean also a possibly contaminated validation or test sample, in which case the optimally hyperparametrized OD model is applied to the possibly contaminated test inputs X_{ts}.

Most of the datasets used in this work do not have a specific test subset. Because of this, to test the different models a fivefold nested Cross Validation (CV) is used, in which first five outer folds are built, and we cyclically use four of them as train-validation subsets for model hyperparameterization and the remaining one for testing. In turn, model hyperparametrization is done again by five fold CV on the train-validation subsets. The optimal hyperparameters are obtained by a grid search; the hyperparameter values used for each model are shown in Table 1. As it can be seen, all the OD models require the choice of a contamination parameter; when used with a linear supervised model M this contamination fraction starts at 5% of the total sample size, as shown in the Table; when the supervised model is a multilayer perceptron, an extra 1% is added to the contamination values in the Table. Also, k denotes the number of neighbors a model uses; ρ in the `pca` method is the fraction of the total number of features used for principal components and α in `hbos` is its internal regularization parameters. The scoring used for optimal hyperparameter selection are the Mean Absolute Error (MAE) for regression and the accuracy for classification; the test MAE or accuracy values will be reported after averaging them over the five outer test folds; their standard deviations will also be given.

4.2. Datasets

We will work with eight regression and eight classification datasets. More precisely, the following datasets will be considered: `abalone`, `boston`, `cal-housing`, `concrete`, `cpusmall`, `mg`, `winequality_red` and `winequality_white` for regression, and `australian`, `breast_cancer`, `diabetes`, `digits`, `dna`, `german`, `satimage` and `segment` for classification. Of these last ones, `dna` has three classes, `satimage` has six, `segment` has seven and `digits` has ten; the other datasets have two classes. More details are given about them below.

- **abalone.** The problem here is to predict the age of abalones (a kind of mollusk) from features involving physical measurements; we will use its version on the LIBSVM data repository [1].
- **boston.** The goal is to estimate the median price of houses in several Boston areas; the subset is preloaded in the *scikit-learn* library [2].
- **cal-housing.** The goal here is to predict median house prices for California districts; we have downloaded it from the Kaggle repository [3].
- **concrete.** We want to predict the compressive strength of concrete variants from a number of physical and chemical measurements; we have downloaded it from the UCI repository [4].
- **cpusmall.** The goal is to predict a computer system activity from several system performance measures; we will use its version on the LIBSVM data repository.
- **mg.** The goal is to predict the next value of the Mackey–Glass time series from the previous six values; we will use its version on the LIBSVM data repository.
- **winequality_red.** We want to model a red wine quality score from a series of physicochemical test measurements. The dataset is downloaded from the Kaggle repository.
- **winequality_white.** This is a variant of the previous problem where we now work with white wine. The dataset is also downloaded from the Kaggle repository.
- **australian.** The goal is to decide whether or not an application is credit-worthy; we will use its version on the LIBSVM data repository.
- **breast_cancer.** The goal is here to predict whether a patient is to be diagnosed with cancer; we will also use its version on the LIBSVM data repository.
- **diabetes.** The objective here is to diagnose the presence of diabetes on a sample of Pima Indian women; we will also use its version on the LIBSVM data repository.
- **digits.** We want to classify pixel rasters as one of the digits from 0 to 9; the subset is preloaded in the *scikit-learn* library.
- **dna.** The goal is to classify splice-junction gene sequences into three different classes; we have downloaded it from the LIBSVM data repository.
- **german.number.** This is another problem where patterns are to be classified as either good or bad credits; we will use its version on the LIBSVM data repository.
- **satimage.** The goal is to classify the central pixel in a satellite image; we will also use here its version on the LIBSVM data repository, working only with the 4,435 training subsample.
- **segment.** We want to classify satellite images into one of seven categories; we will also use here the version on the LIBSVM data repository.

In Table 2, their sample sizes, dimensions and the number of classes for the classification ones are given.

4.3. Regression results

As mentioned, we will consider two linear models, Ridge (RR) and Huber (HR) Regression, as well as a MultiLayer Perceptron Regressor (MLPR) with a single hidden layer with 20 units. As mentioned before, a more complex MLPR model is likely to result in a better performance but given the relatively small sample sizes

[1] LIBSVM Data Repository: https://www.csie.ntu.edu.tw/cjlin/libsvm-tools/datasets/, December 2021

[2] *scikit-learn* Data Repository: https://scikit-learn.org/stable/datasets/toy_dataset.html, December 2021

[3] Kaggle Data Repository: https://www.kaggle.com/datasets?fileType=csv, December 2021

[4] UCI Data Repository: https://archive.ics.uci.edu/ml/index.php, December 2021

Table 1
Set of hyperparameter values for the OD models considered.

Model	Parameter	Range
Logistic Regression	C	$\left[10^{-3}, 10^{-2}, \ldots, 10^{3}\right]$
Ridge Regression	α	$\left[10^{-3}, 10^{-2}, \ldots, 10^{3}\right]$
MCD	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	ρ	$[0.7, 0.8, 0.9, 0.95]$
LOF	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	k	$\{5, 9, 17, 33, 65\}$
iForest	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	num_iTrees	$\{25, 50, 75, 100\}$
	max_features	$\{0.25, 0.5, 0.75, 1\}$
One-class SVM	ν	$\{0.05, 0.1, 0.15, 0.20, 0.25\}$
	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4, 0.5\}$
COF	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	k	$\{5, 9, 17, 33, 65\}$
KNN	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	k	$\{5, 9, 17, 33, 65\}$
HBOS	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	α	$\{0.05, 0.1, 0.2, 0.3, 0.4\}$
SOD	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	k	$\{5, 9, 17, 33, 65\}$
PCA	contamination	$\{0.05, 0.1, 0.15, 0.20, 0.3, 0.4\}$
	ρ	$\{0.5, 0.75, 0.85, 0.9, 0.95\}$

Table 2
Sample sizes, number of features and number of classes.

	size	features	classes
australian	690	14	2
breast-cancer	569	30	2
diabetes	768	8	2
digits	1797	64	10
dna.scale	2000	180	3
german.numer	1000	24	2
satimage.scale	4435	36	6
segment.scale	2310	19	7
abalone	4177	8	-
boston	506	13	-
cal-housing	5000	8	-
concrete	1030	8	-
cpusmall	8192	12	-
mg	1385	6	-
winequality-red	1599	11	-
winequality-white	4898	11	-

and number of dimensions, the architecture used still gives good results, well above those of the linear models. On the other hand, the larger complexity of the MLPRs is bound to give better results than RR and HR; because of this and for a better analysis the individual RR, HR and MLPR will be first reported separately, and then the overall best models will be considered.

Tables 3, 5 and 7 give the mean and standard deviation of the five fold nested RR, HR and MLPR models respectively, while Tables 4, 6 and 8 give for each problems the ranking of the different models by ascending MAE values. As it can be seen, COF gives best results for RR in four problems, SOD and LOF in three, HBOS, KNN and PCA in two and the remaining models, except OCS, in one problem. Notice that the number of best models is larger than the number of problems, as there are MAE ties among several models. The situation changes when Huber regression is considered, where LOF and the base regressor give best results in four problems, COF and OCS in three and all the other models in one; MAE ties happen again in several problems. Notice also that the Huber MAEs are smaller than the RR ones; recall that Huber regression is designed to be robust with respect to target outliers, so the good performance of the base model here is not surprising.

These situations change again for the MLP regressors. Now LOF gives the smallest MAE in three problems (although with a more mediocre performance in the others), HBOS, MCD, PCA IF and the base MLPR give the smallest MAE in two, and the other models in one problem. While the larger MLPR flexibility (and, hence, complexity) leads to good results for the base model, it is nevertheless clear that pipelining it with an OD model results in an enhanced performance. Moreover, it turns out that essentially all MLPR-based models give better results than all the Huber-based ones which, as previously said, give better results than all the Ridge-based ones.

Finally, Table 9 gives the ranking of the six best models selected according to their previously computed rankings across all problems; as it can be seen, these best models use an MLPR as the supervised model and among them HBOS, MCD, IF, PCA and the base MLPR are best in two problems and OCS in one. In any case, notice that these rankings are given only in terms of the average 5-fold MAEs. When the standard deviations are considered, it can

Table 3
5-fold mean and standard deviation of RR models when the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
base	1.592 (0.036)	3.350 (0.292)	0.558 (0.042)	8.272 (0.777)	5.808 (0.300)	0.117 (0.001)	0.504 (0.018)	0.585 (0.007)
cof	1.580 (0.035)	3.231 (0.294)	0.531 (0.052)	8.301 (0.765)	4.696 (0.307)	0.116 (0.001)	0.504 (0.017)	0.586 (0.008)
hbos	1.597 (0.036)	3.362 (0.343)	0.536 (0.054)	8.273 (0.777)	4.736 (0.278)	0.116 (0.001)	0.503 (0.018)	0.586 (0.008)
if	1.607 (0.040)	3.345 (0.295)	0.549 (0.051)	8.233 (0.886)	4.915 (0.533)	0.117 (0.002)	0.503 (0.017)	0.586 (0.008)
knn	1.598 (0.037)	3.237 (0.239)	0.564 (0.085)	8.156 (0.810)	4.674 (0.350)	0.116 (0.002)	0.506 (0.016)	0.586 (0.008)
lof	1.581 (0.043)	3.260 (0.274)	0.537 (0.064)	8.375 (0.859)	5.043 (0.522)	0.116 (0.001)	0.503 (0.016)	0.585 (0.007)
mcd	1.593 (0.033)	3.267 (0.342)	0.567 (0.075)	8.441 (0.979)	4.264 (0.396)	0.117 (0.002)	0.505 (0.014)	0.587 (0.009)
ocs	1.601 (0.036)	3.314 (0.385)	0.569 (0.088)	8.229 (0.888)	5.525 (0.412)	0.117 (0.002)	0.505 (0.013)	0.586 (0.008)
pca	1.601 (0.036)	3.330 (0.378)	0.555 (0.084)	8.377 (1.024)	4.860 (0.320)	0.117 (0.001)	0.503 (0.014)	0.585 (0.008)
sod	1.585 (0.031)	3.235 (0.271)	0.550 (0.087)	8.291 (0.738)	4.644 (0.216)	0.116 (0.001)	0.503 (0.016)	0.585 (0.007)

Table 4
Rankings of the OD models when Ridge regression models are used and the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
sod	3	2	5	6	2	1	1	1	3	2.7
cof	1	1	1	7	4	1	6	5	4	3.3
lof	2	4	3	8	8	1	1	1	3	3.4
hbos	6	10	2	5	5	1	1	5	2	4.1
knn	7	3	8	1	3	1	10	5	2	4.4
if	10	8	4	3	7	6	1	5	1	5.0
pca	8	7	6	9	6	6	1	1	2	5.1
base	4	9	7	4	10	6	6	1	1	5.3
ocs	8	6	10	2	9	6	8	5	0	6.0
mcd	5	5	9	10	1	6	8	10	1	6.1

Table 5
5-fold mean and standard deviation of HR models when the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
base	1.553 (0.036)	3.176 (0.347)	0.533 (0.050)	8.162 (0.828)	4.120 (0.333)	0.115 (0.001)	0.501 (0.017)	0.584 (0.008)
cof	1.554 (0.038)	3.177 (0.332)	0.522 (0.054)	8.197 (0.796)	4.118 (0.330)	0.115 (0.001)	0.502 (0.016)	0.585 (0.008)
hbos	1.555 (0.036)	3.197 (0.348)	0.523 (0.058)	8.182 (0.811)	4.122 (0.334)	0.115 (0.001)	0.503 (0.017)	0.585 (0.008)
if	1.558 (0.037)	3.200 (0.297)	0.541 (0.060)	8.269 (1.000)	4.218 (0.352)	0.116 (0.001)	0.501 (0.015)	0.585 (0.008)
knn	1.556 (0.034)	3.175 (0.319)	0.539 (0.061)	8.157 (0.856)	4.139 (0.327)	0.115 (0.001)	0.503 (0.016)	0.585 (0.009)
lof	1.555 (0.037)	3.168 (0.358)	0.524 (0.059)	8.188 (0.818)	4.118 (0.336)	0.115 (0.001)	0.501 (0.015)	0.585 (0.008)
mcd	1.555 (0.034)	3.206 (0.285)	0.539 (0.056)	8.691 (1.013)	4.219 (0.392)	0.115 (0.001)	0.503 (0.014)	0.586 (0.009)
ocs	1.557 (0.037)	3.188 (0.333)	0.544 (0.063)	8.151 (0.902)	4.220 (0.293)	0.115 (0.001)	0.504 (0.012)	0.584 (0.008)
pca	1.558 (0.037)	3.210 (0.292)	0.534 (0.063)	8.615 (1.034)	4.213 (0.362)	0.115 (0.001)	0.503 (0.014)	0.585 (0.009)
sod	1.554 (0.036)	3.182 (0.334)	0.529 (0.064)	8.204 (0.850)	4.124 (0.342)	0.115 (0.001)	0.502 (0.015)	0.585 (0.008)

Table 6
Rankings of the OD models when Huber regression models are used and the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
base	1	3	5	3	3	1	1	1	4	2.4
lof	4	1	3	5	1	1	1	3	4	2.6
cof	2	4	1	6	1	1	4	3	3	2.8
hbos	4	7	2	4	4	1	6	3	1	3.6
sod	2	5	4	7	5	1	4	3	1	3.6
knn	7	2	7	2	6	1	6	3	1	3.9
ocs	8	6	10	1	10	1	10	1	3	5.6
pca	9	10	6	9	7	1	6	3	1	5.8
if	9	8	9	8	8	10	1	3	1	6.3
mcd	4	9	7	10	9	1	6	10	1	6.3

Table 7
5-fold mean and standard deviation of MLPR models when the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
base	1.515 (0.033)	2.808 (0.321)	0.431 (0.022)	5.378 (0.445)	2.309 (0.024)	0.102 (0.002)	0.492 (0.014)	0.534 (0.018)
cof	1.511 (0.038)	2.656 (0.275)	0.432 (0.021)	5.433 (0.539)	2.329 (0.021)	0.102 (0.002)	0.489 (0.011)	0.542 (0.011)
hbos	1.501 (0.041)	2.558 (0.152)	0.427 (0.011)	5.393 (0.633)	2.309 (0.042)	0.102 (0.001)	0.493 (0.010)	0.540 (0.013)
if	1.495 (0.041)	2.546 (0.157)	0.431 (0.015)	5.592 (0.591)	2.307 (0.052)	0.103 (0.002)	0.494 (0.017)	0.539 (0.013)
knn	1.502 (0.043)	2.673 (0.453)	0.436 (0.016)	5.378 (0.402)	2.362 (0.031)	0.102 (0.001)	0.492 (0.015)	0.544 (0.015)
lof	1.508 (0.047)	2.881 (0.369)	0.427 (0.011)	5.613 (0.321)	2.376 (0.124)	0.102 (0.002)	0.486 (0.018)	0.540 (0.014)
mcd	1.495 (0.047)	2.560 (0.108)	0.436 (0.024)	5.269 (0.529)	2.409 (0.057)	0.102 (0.001)	0.492 (0.018)	0.538 (0.015)
ocs	1.495 (0.041)	2.565 (0.161)	0.439 (0.012)	5.518 (0.435)	2.384 (0.035)	0.102 (0.001)	0.491 (0.014)	0.537 (0.012)
pca	1.493 (0.037)	2.614 (0.190)	0.432 (0.016)	5.712 (1.138)	2.359 (0.030)	0.102 (0.002)	0.491 (0.010)	0.539 (0.019)
sod	1.503 (0.028)	2.714 (0.157)	0.435 (0.020)	5.472 (0.341)	2.313 (0.055)	0.102 (0.001)	0.493 (0.019)	0.538 (0.015)

Table 8
Rankings of the OD models when MLP regression models are used and the test sample is not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
hbos	5	2	1	4	2	1	8	7	2	3.6
base	10	9	3	2	2	1	5	1	2	3.9
mcd	2	3	8	1	10	1	5	3	2	3.9
pca	1	5	5	10	6	1	3	5	2	4.2
ocs	2	4	10	7	9	1	3	2	1	4.3
if	2	1	3	8	1	10	10	5	2	4.7
cof	9	6	5	5	5	1	2	9	1	4.8
sod	7	8	7	6	4	1	8	3	1	5.0
knn	6	7	8	2	7	1	5	10	1	5.2
lof	8	10	1	9	8	1	1	7	3	5.3

Table 9
Rankings of the best six regression models across all datasets when test datasets are not filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	ave	times-best
hbos-mlpr	5	2	1	4	2	1	8	7	3.8	2
mcd-mlpr	2	3	8	1	10	1	5	3	4.1	2
base-mlpr	10	9	3	2	2	1	5	1	4.1	2
pca-mlpr	1	5	5	10	6	1	3	5	4.5	2
ocs-mlpr	2	4	10	7	9	1	3	2	4.8	1
if-mlpr	2	1	3	8	1	10	10	5	5.0	2

Table 10
5-fold mean and standard deviation of RR models when test datasets are filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
cof	1.479 (0.061)	3.562 (0.704)	0.454 (0.012)	9.969 (1.269)	5.507 (0.398)	0.117 (0.002)	0.517 (0.030)	0.573 (0.012)
hbos	1.633 (0.028)	3.254 (0.438)	0.530 (0.014)	7.976 (0.666)	3.942 (0.206)	0.107 (0.005)	0.505 (0.033)	0.574 (0.020)
if	1.514 (0.039)	2.174 (0.344)	0.457 (0.012)	6.025 (0.395)	2.046 (0.076)	0.112 (0.004)	0.455 (0.033)	0.569 (0.010)
knn	1.311 (0.035)	2.641 (0.424)	0.468 (0.014)	8.244 (1.284)	2.680 (0.084)	0.117 (0.004)	0.495 (0.022)	0.579 (0.008)
lof	1.431 (0.047)	3.566 (0.831)	0.454 (0.012)	8.041 (0.634)	5.261 (1.057)	0.122 (0.002)	0.499 (0.034)	0.582 (0.007)
mcd	1.295 (0.036)	2.196 (0.103)	0.443 (0.013)	5.775 (0.517)	1.975 (0.067)	0.116 (0.002)	0.472 (0.021)	0.582 (0.013)
ocs	1.618 (0.115)	2.539 (0.422)	0.451 (0.017)	8.953 (1.995)	2.538 (0.020)	0.126 (0.006)	0.468 (0.034)	0.573 (0.008)
pca	1.642 (0.034)	2.665 (0.553)	0.465 (0.010)	9.597 (0.573)	2.311 (0.057)	0.131 (0.002)	0.498 (0.026)	0.571 (0.007)
sod	1.344 (0.045)	2.993 (0.523)	0.472 (0.024)	7.361 (1.293)	3.065 (0.399)	0.112 (0.013)	0.473 (0.033)	0.576 (0.019)

Table 11
Rankings of the OD models when Ridge regression models are used and the test sample is filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
if	6	1	5	2	2	2	1	1	3	2.6
mcd	1	2	1	1	1	4	3	8	4	2.8
ocs	7	3	2	7	4	8	2	3	0	4.0
sod	3	6	8	3	6	2	4	6	0	4.2
knn	2	4	7	6	5	5	5	7	0	4.6
pca	9	5	6	8	3	9	6	2	0	5.3
hbos	8	7	9	4	7	1	8	5	1	5.6
cof	5	8	3	9	9	5	9	3	0	5.7
lof	4	9	3	5	8	7	7	8	0	5.7

Table 12
5-fold mean and standard deviation of HR models when test datasets are filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
cof	1.434 (0.075)	3.184 (0.731)	0.442 (0.015)	9.780 (1.217)	3.594 (0.509)	0.115 (0.001)	0.509 (0.033)	0.572 (0.014)
hbos	1.611 (0.044)	3.078 (0.397)	0.507 (0.022)	7.634 (0.673)	3.161 (0.257)	0.105 (0.006)	0.495 (0.034)	0.574 (0.020)
if	1.461 (0.045)	2.181 (0.268)	0.448 (0.007)	5.982 (0.466)	1.965 (0.047)	0.109 (0.004)	0.456 (0.026)	0.557 (0.005)
knn	1.249 (0.028)	2.364 (0.211)	0.465 (0.021)	8.174 (1.580)	2.379 (0.104)	0.116 (0.003)	0.499 (0.029)	0.577 (0.008)
lof	1.386 (0.045)	3.222 (0.688)	0.442 (0.014)	7.789 (0.565)	3.947 (0.286)	0.120 (0.002)	0.493 (0.038)	0.580 (0.005)
mcd	1.267 (0.037)	2.096 (0.102)	0.433 (0.013)	5.720 (0.465)	1.901 (0.063)	0.115 (0.002)	0.473 (0.017)	0.582 (0.012)
ocs	1.574 (0.086)	2.505 (0.437)	0.441 (0.020)	9.047 (0.850)	2.471 (0.073)	0.121 (0.004)	0.465 (0.035)	0.577 (0.010)
pca	1.603 (0.036)	3.243 (0.809)	0.451 (0.012)	8.931 (0.762)	2.249 (0.081)	0.124 (0.001)	0.487 (0.020)	0.574 (0.005)
sod	1.302 (0.065)	2.743 (0.571)	0.464 (0.023)	7.183 (1.908)	2.353 (0.235)	0.113 (0.005)	0.481 (0.050)	0.574 (0.022)

Table 13
Rankings of the OD models when Huber regression models are used and the test sample is filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
if	6	2	5	2	2	2	1	1	2	2.6
mcd	2	1	1	1	1	4	3	9	4	2.9
sod	3	5	7	3	4	3	4	3	0	3.6
ocs	7	4	2	8	6	8	2	6	0	4.8
knn	1	3	8	6	5	6	8	6	1	4.9
cof	5	7	3	9	8	4	9	2	0	5.2
hbos	9	6	9	4	7	1	7	3	1	5.2
lof	4	8	3	5	9	7	6	8	0	5.6
pca	8	9	6	7	3	9	5	3	0	5.6

be seen that in almost all problems, a given model mean MAE falls well within the MAE of another plus or minus its standard deviation. In particular, MAE differences across models are not as wide as the rankings may suggest.

As mentioned, the previous results are obtained when the train sample is filtered but not the test one. The same analysis has been repeated when the test datasets are also filtered. Tables 10, 12 and 14 give now the mean and standard deviation in this case of the fivefold nested RR, HR and MLPR models, respectively, and Tables 11, 13 and 15 the model rankings for each problem in ascending MAEs. As it can be seen, when Ridge is the base model, MCD is best in four problems, IF in three, HBOS in one and the others in none. This changes slightly when HR is the base model, with MCD being best again in four problems, IF in two, HBOS and KNN in one and the others in none. MCD is again best in four problems when MLPR is the base model, IF is also best in four problems and COF in one.

In general, it can be said that MCD clearly behaves in a superior manner here, with IF close by. This can also be seen in Table 16 which gives again the ranking of the six best models across all problems. Notice how here the best ranked models are MCD and IF with their combination with an MLPR; they also perform well when combined with a Huber model. Again, the MLPR-based models perform much better than the others. A reason for this different behavior may be the stronger outlier removal that is performed in this case. The average fraction of retained inliers is now 76.3% (i.e., an optimal contamination of about 0.25), while it was 92.9% (i.e., an optimal contamination of about 0.07) when no filtering was done.

4.4. Classification results

Here again most of the classification datasets used do not have a specific test subset and the same 5-fold nested CV described for regression will be applied. The same hyperparameters considered in regression will be used for the OD models, combined with a Logistic Regression (LR) model and a MultiLayer Perceptron Classifier (MLPC) with a single hidden layer with 20 units for classification. For two-class problems the standard formulation of LR will be applied and its multinomial version for multiclass problems; see their web API *scikit-learn* pages for more details. The same L2 regularization hyperparameter range will be used for both supervised models, although their values behave now differently on each model, as the value C used for LR is essentially the inverse of the α value used in the MLPC. Because of this, the ranges considered are inversely symmetric. The scoring used now for optimal hyperparameter selection is the accuracy and the test accuracy values averaged over the five outer test folds and their standard deviations will also be reported.

Tables 17 and 18 give the mean and standard deviation of the five fold nested LR and MLPC models respectively when the test dataset is not filtered. Tables 19 and 20 present the rankings of each OD model for each problem as well as the number of times it is best and its average rankings (notice that there may be again ties of several models on the same problem). As it can be seen, HBOS, LOF and the base model give best results for LR in 2 problems, IF, SOD, COF and MCD give best results in one problem and KNN and OCS are never the best model for any problem. As it hap-

Table 14
5-fold mean and standard deviation of MLPR models when test datasets are filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white
cof	1.481 (0.095)	2.401 (0.298)	0.400 (0.012)	6.997 (3.903)	2.326 (0.060)	0.107 (0.006)	0.490 (0.021)	0.586 (0.082)
hbos	1.640 (0.154)	2.660 (0.192)	0.427 (0.009)	7.731 (1.505)	2.171 (0.105)	0.101 (0.002)	0.474 (0.020)	0.544 (0.019)
if	1.463 (0.038)	2.113 (0.207)	0.407 (0.015)	4.695 (0.364)	1.945 (0.043)	0.097 (0.005)	0.460 (0.025)	0.522 (0.019)
knn	1.398 (0.117)	3.117 (1.006)	0.406 (0.012)	8.925 (4.403)	1.946 (0.048)	0.101 (0.001)	0.487 (0.038)	0.550 (0.038)
lof	1.515 (0.133)	4.113 (1.475)	0.405 (0.012)	6.755 (3.354)	2.228 (0.044)	0.102 (0.001)	0.497 (0.024)	0.564 (0.064)
mcd	1.265 (0.043)	2.188 (0.190)	0.400 (0.011)	4.797 (0.418)	1.895 (0.025)	0.102 (0.003)	0.467 (0.025)	0.521 (0.012)
ocs	1.482 (0.055)	2.446 (0.171)	0.413 (0.009)	5.231 (0.594)	2.245 (0.069)	0.118 (0.016)	0.505 (0.080)	0.538 (0.014)
pca	1.541 (0.047)	2.524 (0.247)	0.406 (0.012)	7.524 (3.468)	2.271 (0.034)	0.110 (0.002)	0.485 (0.016)	0.539 (0.015)
sod	1.324 (0.060)	2.209 (0.543)	0.408 (0.021)	5.389 (0.462)	2.050 (0.067)	0.098 (0.005)	0.484 (0.043)	0.543 (0.033)

Table 15
Rankings of the OD models when MLP regression models are used and the test sample is filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	times-best	ave
mcd	1	2	1	2	1	5	2	1	4	2.1
if	4	1	6	1	2	1	1	2	4	2.4
sod	2	3	7	4	4	2	4	5	0	3.4
knn	3	8	4	9	3	3	6	7	0	4.8
cof	5	4	1	6	9	7	7	9	1	5.4
hbos	9	7	9	8	5	3	3	6	0	5.6
ocs	6	5	8	3	7	9	9	3	0	5.6
pca	8	6	4	7	8	8	5	4	0	5.6
lof	7	9	3	5	6	5	8	8	0	5.7

Table 16
Rankings of the best six regression models across all datasets when test samples are filtered.

	abalone	boston	cal-housing	concrete	cpusmall	mg	wine-red	wine-white	ave	times-best
mcd-mlpr	2	5	1	2	1	5	5	1	2.8	3
if-mlpr	14	2	6	1	3	1	3	2	4.0	2
sod-mlpr	7	7	7	4	8	2	12	5	6.5	0
if-huber	13	4	15	7	5	10	2	8	8.0	0
mcd-huber	3	1	10	5	2	15	8	24	8.5	1
if-ridge	18	3	20	8	7	12	1	10	9.9	1

Table 17
5-fold mean and standard deviation when LR models are used and the test sample is not filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment
base	0.867 (0.025)	0.979 (0.015)	0.769 (0.034)	0.963 (0.002)	0.952 (0.011)	0.758 (0.023)	0.858 (0.004)	0.952 (0.009)
cof	0.865 (0.025)	0.975 (0.014)	0.766 (0.029)	0.964 (0.002)	0.952 (0.008)	0.749 (0.022)	0.855 (0.003)	0.949 (0.005)
hbos	0.875 (0.028)	0.972 (0.016)	0.769 (0.027)	0.962 (0.007)	0.952 (0.011)	0.764 (0.017)	0.860 (0.005)	0.947 (0.008)
if	0.867 (0.025)	0.974 (0.016)	0.775 (0.033)	0.960 (0.004)	0.949 (0.012)	0.758 (0.021)	0.857 (0.004)	0.943 (0.009)
knn	0.867 (0.031)	0.974 (0.014)	0.772 (0.030)	0.962 (0.003)	0.951 (0.010)	0.773 (0.016)	0.854 (0.005)	0.941 (0.007)
lof	0.858 (0.023)	0.970 (0.013)	0.768 (0.034)	0.964 (0.004)	0.954 (0.009)	0.766 (0.034)	0.859 (0.009)	0.949 (0.007)
mcd	0.864 (0.024)	0.975 (0.013)	0.767 (0.030)	0.959 (0.005)	0.949 (0.011)	0.777 (0.017)	0.847 (0.007)	0.941 (0.003)
ocs	0.859 (0.019)	0.974 (0.011)	0.768 (0.021)	0.963 (0.002)	0.950 (0.011)	0.766 (0.026)	0.840 (0.009)	0.942 (0.008)
sod	0.865 (0.029)	0.975 (0.013)	0.762 (0.040)	0.964 (0.006)	0.950 (0.010)	0.760 (0.014)	0.858 (0.010)	0.945 (0.008)

Table 18
5-fold mean and standard deviation when MLPC models are used and the test sample is not filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment
base	0.870 (0.029)	0.972 (0.007)	0.771 (0.022)	0.962 (0.005)	0.952 (0.005)	0.757 (0.028)	0.889 (0.003)	0.942 (0.018)
cof	0.867 (0.019)	0.970 (0.018)	0.771 (0.030)	0.953 (0.018)	0.953 (0.006)	0.764 (0.015)	0.887 (0.008)	0.942 (0.021)
hbos	0.864 (0.027)	0.970 (0.019)	0.760 (0.033)	0.952 (0.020)	0.952 (0.007)	0.761 (0.010)	0.888 (0.009)	0.929 (0.014)
if	0.864 (0.024)	0.965 (0.014)	0.754 (0.043)	0.961 (0.002)	0.952 (0.007)	0.750 (0.015)	0.890 (0.004)	0.938 (0.021)
knn	0.864 (0.017)	0.967 (0.017)	0.736 (0.041)	0.959 (0.010)	0.952 (0.007)	0.754 (0.019)	0.890 (0.006)	0.935 (0.013)
lof	0.864 (0.022)	0.968 (0.015)	0.759 (0.025)	0.963 (0.008)	0.954 (0.007)	0.765 (0.009)	0.890 (0.010)	0.950 (0.014)
mcd	0.874 (0.018)	0.963 (0.013)	0.701 (0.047)	0.965 (0.003)	0.954 (0.008)	0.770 (0.026)	0.888 (0.002)	0.943 (0.014)
ocs	0.855 (0.024)	0.965 (0.014)	0.769 (0.032)	0.963 (0.007)	0.950 (0.010)	0.756 (0.018)	0.884 (0.007)	0.923 (0.032)
sod	0.858 (0.015)	0.956 (0.014)	0.754 (0.037)	0.965 (0.009)	0.949 (0.007)	0.749 (0.021)	0.886 (0.008)	0.942 (0.011)

Table 19
Rankings of the OD models when LR models are used and the test sample is not filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	times-best	ave
base	2	1	3	4	2	7	3	1	2	2.8
hbos	1	8	3	6	2	5	1	4	2	3.6
lof	9	9	5	1	1	3	2	2	2	3.8
cof	5	2	8	1	2	9	6	2	1	4.0
knn	2	5	2	6	5	2	7	8	0	4.1
sod	5	2	9	1	6	6	3	5	1	4.2
if	2	5	1	8	8	7	5	6	1	4.8
ocs	8	5	5	4	6	3	9	7	0	5.2
mcd	7	2	7	9	8	1	8	8	1	5.7

Table 20
Rankings of the OD models when MLPC models are used and the test sample is not filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	times-best	ave
lof	4	4	5	3	1	2	1	1	3	2.7
base	2	1	1	5	4	5	4	3	2	3.0
cof	3	2	1	8	3	3	7	3	1	3.4
mcd	1	8	9	1	1	1	5	2	4	3.6
hbos	4	2	4	9	4	4	5	8	0	4.4
if	4	6	6	6	4	8	1	6	1	4.7
knn	4	5	8	7	4	7	1	7	1	4.9
ocs	9	6	3	3	8	6	9	9	0	5.9
sod	8	9	6	1	9	9	8	3	1	6.0

Table 21
Rankings of the best six classification models across all datasets when the test sample is not filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	ave	times-best
base-logregr	4	1	5	6	5	11	12	1	5.6	2
hbos-logregr	1	8	5	10	5	7	10	5	6.4	1
lof-mlpc	10	13	14	6	1	6	1	2	6.6	2
mcd-mlpc	2	17	18	1	1	3	5	7	6.8	2
base-mlpc	3	8	3	10	5	13	4	9	6.9	0
lof-logregr	16	10	8	3	1	4	11	3	7.0	1

pened in regression, the situation changes for the MLPC, for which MCD gives the largest accuracy in four problems, LOF in three, the base model in two, SOD, IF, KNN and COF in one, and HBOS and One-class SVM in none. Thus, it is again clear that for both the LR and MLPC, pipelining a supervised classifier with an OD model leads to an enhanced performance.

Considering together the LR and MLPC model results, Table 21 gives the ranking of the six best models across all problems, where the best is to be understood as having a smallest ranking average. As it can be seen, LOF-MLPC, MCD-MLPC and, slightly surprising, the base LR model outperform on two problems. A possible explanation for this is that the MLPC-based models seem to perform worse on the `breast-cancer` and `segment` problems. Also, the base LR model has a balanced performance across all datasets while, for instance, MCD-MLPC is the worst model for `diabetes` and the worst by one for `breast-cancer`; this penalizes its average rankings. Notice also that there are some ranking ties but fewer than in the case of regression. Finally, observe that the average number of inliers retained is 92.7%, i.e., an average optimal contamination of about 0.073.

When the test sample is filtered, things change again, as now it can be seen in Table 22 that MCD is best in three problems when combined with LR, COF is best in two, HBOS, IF, KNN, LOF and SOD are best in one, and One-class SVM in none. This can be checked in Table 24 which shows the rankings of each OD-LR model combination for each problem as well as the number of times it is best and its average rankings. When the OD models are combined with an MLPC, Table 23 shows that LOF and COF are best in two datasets, IF, KNN, MCD and One-class SVM are the best in one, and HBOS and SOD in none; this can be checked now in Table 25 with the rankings of each OD plus an MLPC model for each problem. Table 26, which gives again the six best models across all problems ordered by their averaging rankings across all datasets, also confirms this change. Now COF-LR is best in two problems (but with mediocre rankings in some others) and five of the best models are based on LR; the only MLPC model entering the tables is MCD. But we point out again that these results are given only in terms of rankings of the average accuracies and, for both LR and MLPCs, these model accuracies fall in almost all problems within the accuracy of another model plus or minus its standard deviation. Finally, the average fraction of retained inliers is now 88.6%, slightly smaller than that fraction when not filtering was done.

It can be also concluded here that pipelining an OD model with an LR or MLPC model leads in general to a performance better than that of standalone LR or MLPCs. On the other hand, filtering the test sample doesn't ensure a better performance. In fact, for classification this is only the case for the `diabetes`, `german` and `satimage` problems, although in all cases the accuracy differences are rather small. Finally, it is difficult to single out best OD models but when the test sample is not filtered, LOF performs consistently on top; when the test sample is filtered, MCD and COF stand slightly above the rest but here LOF also performs well.

4.5. Execution Time Considerations

Finally, the relative execution times of the OD models will be briefly examined. There are two main contributions to execution times, that of building the OD method and that of training the corresponding supervised classifier or regressor. Of course, these times will also be affected by the concrete R_α, R_β OD and supervised hyperparameters each method uses. In order to have a comprehen-

Table 22
5-fold mean and standard deviation when LR models are used and the test sample is filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment
cof	0.832 (0.048)	0.968 (0.015)	0.788 (0.046)	0.975 (0.006)	0.932 (0.021)	0.722 (0.028)	0.895 (0.011)	0.955 (0.012)
hbos	0.871 (0.010)	0.945 (0.046)	0.773 (0.031)	0.962 (0.005)	0.939 (0.015)	0.766 (0.043)	0.853 (0.005)	0.944 (0.008)
if	0.866 (0.018)	0.972 (0.012)	0.788 (0.042)	0.966 (0.010)	0.943 (0.009)	0.782 (0.023)	0.856 (0.009)	0.951 (0.010)
knn	0.842 (0.045)	0.971 (0.021)	0.779 (0.019)	0.949 (0.015)	0.941 (0.012)	0.772 (0.020)	0.919 (0.008)	0.935 (0.013)
lof	0.856 (0.047)	0.975 (0.018)	0.784 (0.028)	0.969 (0.023)	0.938 (0.032)	0.763 (0.022)	0.893 (0.019)	0.955 (0.012)
mcd	0.866 (0.016)	0.979 (0.010)	0.786 (0.028)	0.977 (0.005)	0.976 (0.010)	0.784 (0.037)	0.884 (0.011)	0.945 (0.008)
ocs	0.868 (0.019)	0.966 (0.019)	0.782 (0.017)	0.968 (0.008)	0.939 (0.017)	0.770 (0.021)	0.863 (0.007)	0.879 (0.048)
sod	0.870 (0.016)	0.967 (0.015)	0.776 (0.035)	0.978 (0.021)	0.950 (0.006)	0.780 (0.033)	0.907 (0.010)	0.954 (0.021)

Table 23
5-fold mean and standard deviation when MLPC models are used and the test sample is filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment
cof	0.867 (0.023)	0.980 (0.010)	0.732 (0.045)	0.966 (0.007)	0.932 (0.007)	0.739 (0.024)	0.901 (0.015)	0.945 (0.010)
hbos	0.862 (0.029)	0.968 (0.011)	0.732 (0.061)	0.940 (0.016)	0.933 (0.012)	0.728 (0.022)	0.883 (0.013)	0.930 (0.031)
if	0.867 (0.015)	0.966 (0.026)	0.755 (0.050)	0.959 (0.012)	0.955 (0.009)	0.786 (0.021)	0.883 (0.008)	0.937 (0.025)
knn	0.860 (0.025)	0.937 (0.081)	0.768 (0.015)	0.907 (0.120)	0.932 (0.009)	0.751 (0.038)	0.884 (0.007)	0.934 (0.013)
lof	0.876 (0.021)	0.939 (0.075)	0.733 (0.050)	0.917 (0.106)	0.943 (0.011)	0.753 (0.030)	0.905 (0.009)	0.935 (0.023)
mcd	0.871 (0.014)	0.970 (0.011)	0.767 (0.032)	0.961 (0.004)	0.959 (0.015)	0.781 (0.021)	0.897 (0.008)	0.938 (0.013)
ocs	0.810 (0.103)	0.982 (0.011)	0.758 (0.016)	0.957 (0.005)	0.673 (0.232)	0.766 (0.008)	0.863 (0.021)	0.924 (0.037)
sod	0.871 (0.022)	0.970 (0.008)	0.752 (0.059)	0.956 (0.022)	0.923 (0.059)	0.775 (0.019)	0.899 (0.017)	0.932 (0.014)

Table 24
Rankings of the OD models when LR models are used and the test sample is filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	times-best	ave
mcd	4	1	3	2	1	1	5	5	3	2.8
sod	2	6	7	1	2	3	2	3	1	3.0
if	4	3	1	6	3	2	7	4	1	3.4
lof	6	2	4	4	7	7	4	1	1	4.0
cof	8	5	1	3	8	8	3	1	2	4.3
knn	7	4	6	8	4	4	1	7	1	4.7
ocs	3	7	5	5	5	5	6	8	0	4.9
hbos	1	8	8	7	5	6	8	6	1	5.6

Table 25
Rankings of the OD models when MLPC models are used and the test sample is filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	times-best	ave
mcd	2	3	2	2	1	2	4	2	1	2.1
if	4	6	4	3	2	1	6	3	1	3.3
cof	4	2	7	1	5	7	2	1	2	3.4
sod	2	3	5	5	7	3	3	6	0	3.8
lof	1	7	6	7	3	5	1	4	2	4.0
ocs	8	1	3	4	8	4	8	8	1	5.0
knn	7	8	1	8	5	6	5	5	1	5.1
hbos	6	5	7	6	4	8	6	7	0	5.4

Table 26
Rankings of the best six classification models across all datasets when the test sample is filtered.

	australian	breast-cancer	diabetes	digits	dna	german	satimage	segment	ave	times-best
mcd-logregr	9	3	3	2	1	2	9	5	4.2	1
sod-logregr	5	11	7	1	4	5	2	3	4.8	1
if-logregr	9	5	1	6	5	3	15	4	6.0	1
mcd-mlpc	2	7	10	9	2	4	6	8	6.0	0
lof-logregr	13	4	4	4	10	11	8	1	6.9	1
cof-logregr	15	9	1	3	12	16	7	1	8.0	2

Table 27
Number and names of parameters on the OD methods considered. A symbol - means that the model in question does not use the parameter.

	cont	k	rho	alpha	nu	max-feat	num-iTrees	C	all
cof	6	5	-	-	-	-	-	7	210
hbos	6	-	-	5	-	-	-	7	210
if	6	-	-	-	-	4	4	7	672
knn	6	5	-	-	-	-	-	7	210
lof	6	5	-	-	-	-	-	7	210
mcd	6	-	4	-	-	-	-	7	168
ocs	6	-	-	-	5	-	-	7	210
pca	6	-	5	-	-	-	-	7	210
sod	6	5	-	-	-	-	-	7	210

sive evaluation of these effects, the overall hyperparametrization times will be measured on a cross validation scenario or, in other words, the times measured are those of training the pairing of an OD and a supervised model over a rather large number of different hyperparameter value combinations. This has been done under the same 5-fold CV setting used in the previous experiments. The total numbers $|R_\alpha \times R_\beta|$ of these combinations are shown in column `all` of Table 27; it has as columns the hyperparameter names ("-" in a hyperparameter name column means that the OD model in question does not use it). As it can be seen in the table, all methods but IF have fairly similar numbers of hyperparameters (notice that IF has three hyperparameters, while the other OD methods have two). Thus, the overall hyperparameterization times have been normalized by dividing them by the number of hyperparameters each model combination has, as shown in Table 27.

Since the goal here is to emphasize the contribution of the OD methods to the overall time, linear and logistic regression will be used as regressors and classifiers, respectively, as they have the shortest training times among the supervised methods considered and, hence, should result in a heavier weight being given to the OD models in the overall times. Total hyperparameter times have been measured for all the classification and regression datasets; moreover, and to have relative times, for each dataset, all normalized hyperparametrization times are divided by the smallest normalized time, which is that of PCA -based OD in all problems but `dna` (for which the fastest method is LOF) and `segment` (for which it is HBOS). We have then computed the averages of these relative times over all datasets and then sorted the OD methods by increasing relative times. We point out that the reported values are wall clock times measured during continuous execution on a slurm [5] batch queue. In particular, they cannot be taken as precise timings for the OD methods involved as, besides classifier and regressor times, they also include, for instance, dataset loading times; however, these affect equally the faster and slower methods, so our relative times mitigate their impact. Notice also that while batch queue processing guarantees in principle uninterrupted execution, wall times can be affected by system interruptions, which should increase the longer a queue takes to process a given job. Finally, we have used the *scikit-learn* and *PyOD* libraries on an "as is" basis, without delving on any details of their Python implementations which, given the processing overheads due Python's interpreted nature, can substantially influence execution speed.

Because of this, the following time values have to be taken more as ballpark estimates than as precise measures. However, and on the other hand, they not only give a good idea of the order of magnitudes involved in the underlying methods and implementations, but are also quite representative of the actual times the practitioner has to expect when hyperparametrizing the OD models following the approach proposed here.

Fig. 1 shows these relative times with no filtering of the test sample (similar results are obtained when test filtering is done). As it can be seen, four different regimes can be distinguished:

- A low computational cost regime involving the PCA and HBOS models.
- A medium cost one involving LOF, IF and KNN, with relative averages approximately between 5 and 15 times that of PCA.

[5] slurm Workload Manager: https://slurm.schedmd.com/quickstart.html, December 2021

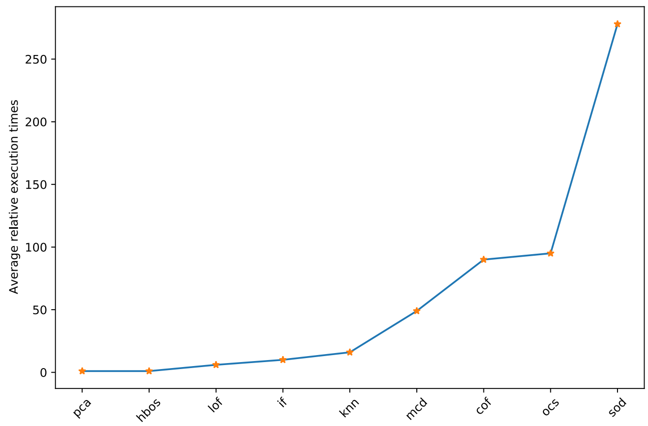


Fig. 1. Averages of relative execution times for all OD. Test samples are not filtered.

- A large cost one involving MCD, COF and OCS, whose relative costs can be approximately about between 50 (form MCD) and 100 (for COF and OCS) times those of PCA.
- Finally, a very high computational regime associated to SOD.

Even with the above caveats, these costs can be used to put the previous performance results in perspective. For instance, we have seen for classification that, when no test filtering is done, HBOS, LOF and MCD appear to have the best overall accuracies, while for regression HBOS, MCD, IF and even PCA have the better performances. Thus, in the context of the just discussed execution times, it can be said that HBOS would be the best approach, as MCD requires much longer execution times; on the other hand, LOF, IF and PCA, not far behind in performance, have the plus of relatively short execution times, particularly in the case of PCA.

However, these times may vary substantially with the dataset sizes, as shown in Table 28, which gives for what we can consider the best performers in the experiments, i.e. PCA, HBOS, LOF, IF and MCD, their relative times over the different datasets rounded to the nearest integer and, for convenience, their sample sizes; the datasets have been sorted by increasing sample sizes. As it can be seen, MCD relative times are in most cases up to 5 times larger than those of IF, with the large exception of the `dna` dataset, where MCD's time shots up to about 100 times that of IF. In this same vein, IF times are in most cases less than eight times larger than those of LOF (and even smaller for some datasets), with HBOS and PCA being consistently the fastest. This suggests that preferred OD methods for many problems could be MCD and IF, although MCD may require a substantial computational effort for some of them. On the other hand, PCA, HBOS and LOF appear to be always worth trying, as they have a fast execution.

5. Conclusions and further work

In this work, a new supervised framework for optimal hyperparameter estimation of Outlier Detectors (OD) has been proposed. The main goal has been to overcome the difficulty of choosing the optimal values of the hyperparameters on which OD models rely, a difficulty due to their unsupervised nature. To reach that goal, the new method consists on the combination of an outlier detector and a supervised model, defining a new joint estimator with a pipeline structure. The advantage of this estimator is that it allows to use the classification or regression targets to supervise the search of the best hyperparameters for each detector in an automatic and objective way. Moreover, a second advantage of our approach is to improve in many problems the performance that can be achieved by the classifier or regressor alone.

In this work, nine different OD methods have been considered jointly with three different regressors and two classifiers. The experiments have been carried out over eight different classification and regression datasets. The obtained results allow to conclude that the supervised framework proposed can be very successful when selecting the best detector hyperparameters, as it can be seen in the results summarized in Tables 9, 16, 21 and 26. Moreover, these tables also allow a global evaluation of the OD models used. In fact, while the selection of an outlier detector is a problem-dependent task for which there are many diverse proposals in the literature under quite different hypotheses, our supervised pipeline proposal defines a common ground where competing OD models can be compared, not only in terms of classification and regression performance but also in terms of execution times, as Fig. 1 shows for all the OD methods and Table 28 shows for the five OD methods that appear to have a better performance. In this respect, and regarding the results obtained, we can conclude that, in general, the MCD and IF models stand out as a good option in general, while PCA, HBOS and LOF should also be considered when computational times are a factor to be weighted in.

In any case, and while we have compared a wide number of outlier detectors, they can be considered by now as "classical"

Table 28
Relative times of best OD models over the considered datasets as well as their sizes.

	pca	hbos	lof	if	mcd	size
boston	1	1	2	16	8	506
breast-c	1	1	1	7	17	569
australian	1	1	2	13	7	690
diabetes	1	1	2	13	6	768
german	1	1	3	8	57	1000
concrete	1	1	3	17	60	1030
mg	1	1	2	16	66	1385
wine-red	1	1	7	13	57	1599
digits	1	1	2	3	14	1797
dna	2	1	1	2	224	2000
segment	1	1	3	3	11	2310
abalone	1	1	7	9	53	4177
satimage	1	1	2	2	19	4435
wine-white	1	1	18	10	51	4898
cal-housing	1	1	15	13	70	5000
cpusmall	1	1	28	12	63	8192

approaches in a field where new contributions are constantly appearing. This is in particular the case with the new deep neural network proposals that have been given for both OD and ML modeling; moreover, these new methods allow for the flexible definition and easy application of non standard model losses, which suggest using these deep architectures to couple the OD and modeling components in a pipeline in a more tight way than what is possible under the OD models considered here. These are clear paths with which to continue this line of study.

CRediT authorship contribution statement

Ángela Fernández: Methodology, Software, Investigation, Writing - original draft, Writing - review & editing. **Juan Bella:** Methodology, Software, Investigation, Data curation. **José R. Dorronsoro:** Conceptualization, Software, Methodology, Validation, Investigation, Supervision, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors acknowledge financial support from the European Regional Development Fund and the Spanish State Research Agency of the Ministry of Economy, Industry, and Competitiveness under the projects TIN2016-76406-P (AEI/FEDER, UE) and PID2019-106827GB-I00. They also thank the UAM–ADIC Chair for Data Science and Machine Learning and gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

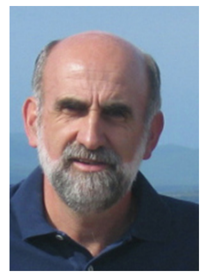
References

[1] D. Hawkins, Identification of outliers, Monographs on applied probability and statistics, Chapman and Hall, 1980.
[2] C.C. Aggarwal, Outlier Analysis, Springer, 2013.
[3] scikit learn developers, Novelty and outlier detection, https://scikit-learn.org/stable/modules/outlier_detection.html (2019).
[4] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58.
[5] V.J. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2) (2004) 85–126.
[6] H. Kriegel, P. Kröger, E. Schubert, A. Zimek, Interpreting and unifying outlier scores, in: Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28–30, 2011, Mesa, Arizona, USA, SIAM/ Omnipress, 2011, pp. 13–24.
[7] Z. Niu, S. Shi, J. Sun, X. He, A survey of outlier detection methodologies and their applications, in: Artificial Intelligence and Computational Intelligence - Third International Conference, AICI 2011, Taiyuan, China, September 24–25, 2011, Proceedings, Part I, Vol. 7002 of Lecture Notes in Computer Science, Springer, 2011, pp. 380–387.
[8] K. Noto, C.E. Brodley, D.K. Slonim, Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection, Data Min. Knowl. Discov. 25 (1) (2012) 109–133.
[9] H. Paulheim, R. Meusel, A decomposition of the outlier detection problem into a set of supervised learning problems, Mach. Learn. 100 (2–3) (2015) 509–531.
[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[11] Y. Zhao, Z. Nasrullah, Z. Li, Pyod: A python toolbox for scalable outlier detection, J. Mach. Learn. Res. 20 (96) (2019) 1–7.
[12] C.C. Aggarwal, S. Sathe, Outlier Ensembles - An Introduction, Springer, 2017.
[13] K. Yamanishi, J. Takeuchi, G.J. Williams, P. Milne, On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms, Data Min. Knowl. Discov. 8 (3) (2004) 275–300.
[14] S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: A comparative evaluation, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24–26, 2008, Atlanta, Georgia, USA, SIAM, 2008, pp. 243–254.
[15] J. Allan, R. Papka, V. Lavrenko, On-line new event detection and tracking, SIGIR Forum 51 (2) (2017) 185–193.
[16] S. Shekhar, C. Lu, P. Zhang, A unified approach to detecting spatial outliers, GeoInformatica 7 (2) (2003) 139–166.
[17] M. Gupta, J. Gao, C.C. Aggarwal, J. Han, Outlier Detection for Temporal Data, Morgan & Claypool Publishers, Synthesis Lectures on Data Mining and Knowledge Discovery, 2014.
[18] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Min. Knowl. Discov. 29 (3) (2015) 626–688.
[19] C.C. Aggarwal, S. Sathe, Theoretical foundations and algorithms for outlier ensembles, SIGKDD Explor. 17 (1) (2015) 24–47.
[20] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58.
[21] P.J. Rousseeuw, K.V. Driessen, A fast algorithm for the minimum covariance determinant estimator, Technometrics 41 (3) (1999) 212–223, https://doi.org/10.1080/00401706.1999.10485670.
[22] J. Hardin, D.M. Rocke, Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator, Comput. Stat. Data Anal. 44 (4) (2004) 625–638, https://doi.org/10.1016/S0167-9473(02)00280-3.
[23] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, ACM SIGMOD Record 29 (2) (2000) 93–104, https://doi.org/10.1145/335191.335388.
[24] J. Tang, Z. Chen, A.W.-C. Fu, D.W. Cheung, Enhancing effectiveness of outlier detections for low density patterns, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2002, pp. 535–548.
[25] F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in: European conference on principles of data mining and knowledge discovery, Springer, 2002, pp. 15–27.
[26] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 427–438.
[27] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, https://doi.org/10.1109/icdm.2008.17.
[28] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discovery Data 6 (1) (2012) 1–39, https://doi.org/10.1145/2133360.2133363.
[29] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471, https://doi.org/10.1162/089976601750264965.
[30] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, A novel anomaly detection scheme based on principal component classifier, Tech. rep., Miami Univ. Coral Gables FL. Dept. of Electrical and Computer Engineering (2003).
[31] C.C. Aggarwal, Outlier analysis, in: Data mining, Springer, 2015, pp. 237–263.
[32] H.-P. Kriegel, P. Kröger, E. Schubert, A. Zimek, Outlier detection in axis-parallel subspaces of high dimensional data, in: Pacific-asia conference on knowledge discovery and data mining, Springer, 2009, pp. 831–838.
[33] M. Goldstein, A. Dengel, Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm, KI-2012: Poster and Demo Track (2012) 59–63.
[34] H. Kriegel, M. Schubert, A. Zimek, Angle-based outlier detection in high-dimensional data, in: Y. Li, B. Liu, S. Sarawagi (Eds.), Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24–27, 2008, ACM, 2008, pp. 444–452.
[35] Z. Li, Y. Zhao, N. Botta, C. Ionescu, X. Hu, COPOD: copula-based outlier detection, in: C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, X. Wu (Eds.), 20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020, IEEE, 2020, pp. 1118–1123.
[36] Y. Zhao, M.K. Hryniewicki, XGBOD: improving supervised outlier detection with unsupervised representation learning, in: 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8–13, 2018, IEEE, 2018, pp. 1–8.
[37] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: R. Grossman, R.J. Bayardo, K.P. Bennett (Eds.), Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21–24, 2005, ACM, 2005, pp. 157–166.
[38] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, CoRR abs/1901.03407.
[39] M.S. Minhas, J.S. Zelek, Semi-supervised anomaly detection using autoencoders, CoRR abs/2001.03674.
[40] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13–17, 2017, ACM, 2017, pp. 665–674.
[41] Z. Chen, C.K. Yeo, B. Lee, C.T. Lau, Autoencoder-based network anomaly detection, in: 2018 Wireless Telecommunications Symposium, WTS 2018, Phoenix, AZ, USA, April 17-20, 2018, IEEE, 2018, pp. 1–5.
[42] Y. Ma, P. Zhang, Y. Cao, L. Guo, Parallel auto-encoder for efficient outlier detection, in: Proceedings of the 2013 IEEE International Conference on Big Data, IEEE Computer Society, Santa Clara, CA, USA, 2013, pp. 15–17.
[43] S. Ger, D. Klabjan, Autoencoders and generative adversarial networks for anomaly detection for sequences, CoRR abs/1901.02514.

[44] L. Beggel, M. Pfeiffer, B. Bischl, Robust anomaly detection in images using adversarial autoencoders, in: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Vol. 11906 of Lecture Notes in Computer Science, Springer, 2019, pp. 206–222.
[45] F.D. Mattia, P. Galeone, M.D. Simoni, E. Ghelfi, A survey on gans for anomaly detection, CoRR abs/1906.11632.
[46] P. Oza, V.M. Patel, One-class convolutional neural network, IEEE Signal Process. Lett. 26 (2) (2019) 277–281.
[47] J.H.M. Janssens, I. Flesch, E.O. Postma, Outlier detection with one-class classifiers from ML and KDD, in: International Conference on Machine Learning and Applications, ICMLA 2009, IEEE Computer Society, Miami Beach, Florida, USA, 2009, pp. 147–153.
[48] L. Swersky, H.O. Marques, J. Sander, R.J.G.B. Campello, A. Zimek, On the evaluation of outlier detection and one-class classification methods, in: 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada, October 17–19, 2016, IEEE, 2016, pp. 1–10.
[49] X. Gu, L. Akoglu, A. Rinaldo, Statistical analysis of nearest neighbor methods for anomaly detection, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada, 2019, pp. 10921–10931.
[50] R. Kaur, S. Jha, A. Roy, O. Sokolsky, I. Lee, Are all outliers alike? on understanding the diversity of outliers for detecting oods, arXiv e-prints arXiv:2103.12628.

Ángela Fernández is Assistant Professor at Universidad Autónoma de Madrid (UAM), where she is a member of the Machine Learning Group. She got her PhD in Computer Science from UAM, where she also got a master's degree in Computer Science and the bachelor degrees of Computer Engineering and Mathematics.

Juan Bella is a Computer Engineer by the Universidad Autónoma de Madrid who has just got a master's degree in Computer Science at UAM. He is currently working as a data analyst.



José R. Dorronsoro is Professor of Computer Engineering at Universidad Autónoma de Madrid and a senior scientist at the Instituto de Ingeniería del Conocimiento. He has authored more than 100 scientific papers, has managed a large number of projects and has 10 years' experience on wind and solar energy prediction.