Imbalanced regression using regressor-classifier ensembles

Downloaded from: https://research.chalmers.se, 2026-09-14 09:29 UTC

Citation for the original published paper (version of record):
Orhobor, O., Grinberg, N., Soldatova, L. et al (2023). Imbalanced regression using regressor-classifier ensembles. Machine Learning, 112(4): 1365-1387.
http://dx.doi.org/10.1007/s10994-022-06199-4

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Machine Learning
https://doi.org/10.1007/s10994-022-06199-4

Imbalanced regression using regressor-classifier ensembles

Oghenejokpeme I. Orhobor[1] · **Nastasiya F. Grinberg**[2,3] · **Larisa N. Soldatova**[4] · **Ross D. King**[1,5,6]

Received: 1 March 2021 / Revised: 13 May 2022 / Accepted: 26 May 2022
© The Author(s) 2022

Abstract
We present an extension to the federated ensemble regression using classification algorithm, an ensemble learning algorithm for regression problems which leverages the distribution of the samples in a learning set to achieve improved performance. We evaluated the extension using four classifiers and four regressors, two discretizers, and 119 responses from a wide variety of datasets in different domains. Additionally, we compared our algorithm to two resampling methods aimed at addressing imbalanced datasets. Our results show that the proposed extension is highly unlikely to perform worse than the base case, and on average outperforms the two resampling methods with significant differences in performance.

Keywords Ensemble regression · Machine learning · Imbalanced data

1 Introduction

Class imbalance (Ali et al., 2015; Japkowicz & Stephen, 2002) is a primary concern when building classifiers, where the under-representation of one class can lead to sub-optimal predictive accuracy on future samples. In our previous work (Orhobor et al.,

Editors: Annalisa Appice, Grigorios Tsoumakas.

✉ Oghenejokpeme I. Orhobor
oghenejokpeme.orhobor@gmail.com

[1] Department of Chemical Engineering and Biotechnology, University of Cambridge, CB3 0AS Cambridge, UK

[2] Cambridge Institute of Therapeutic Immunology & Infectious Disease, Jeffrey Cheah Biomedical Centre, Department of Medicine, University of Cambridge, Cambridge Biomedical Campus, CB2 0AW Cambridge, UK

[3] NIAB, 93 Lawrence Weaver Road, CB3 0LE Cambridge, UK

[4] Department of Computing, Goldsmiths, University of London, SE14 6NW London, UK

[5] The Alan Turing Institute, NW1 2DB London, UK

[6] Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden

Published online: 27 June 2022

2020), we argued that the same can be said for the distribution of a response one is interested in modelling for regression problems. The idea is that for a given response, certain parts of the distribution might be underrepresented, making it difficult for a model built using this data to make accurate predictions for unseen samples falling in the underrepresented regions. To this end, we proposed an ensemble learning approach which takes the underlying distribution into account.

Assume we have some learning data, a discretizer of choice, and a bin size of n. We build a regressor for the base case, where $n = 1$. For all other values of n where $n > 1$, and up to some maximum bin size, we split the learning data into n bins using the discretizer. We then build a regressor for each bin in each of the n independent subsets, and a classifier which predicts the probability that a new sample belongs in each of the bins. This classifier is used to combine the predictions from the independent regressors given an unseen sample. Given some unseen data, for each bin size n the regression models produce n predictions, one for each bin. These are combined using bin probabilities provided by the corresponding classifier. Finally, these combined predictions are aggregated using weights that are determined on the learning set. The intuition is that this process should leverage the predictive accuracy of the base and binned models, significantly reducing the possibility of performing worse than the base case, irrespective of the choice of discretizer, which improves upon our previous work (Orhobor et al., 2020). The proposed algorithm is described in Fig. 1 and is formalised in Sect. 3. We demonstrate empirically that it is highly unlikely that the proposed algorithm will perform worse than the base case.

In the next section, we discuss the different techniques comprising the different parts of the proposed extension and the algorithm in general. In Sect. 3 we describe the new algorithm in detail. The evaluation setup is described in Sect. 4 and the results are presented in Sect. 5. We discuss possible improvements to the proposed algorithm and conclude in Sect. 6.

2 Background

Ensemble learning, that is, the aggregation of the predictions made by multiple learning algorithms plays a crucial role in both the original algorithm and the extension we propose. Specifically, we use ensembles in (i) aggregating the predictions made within a given bin

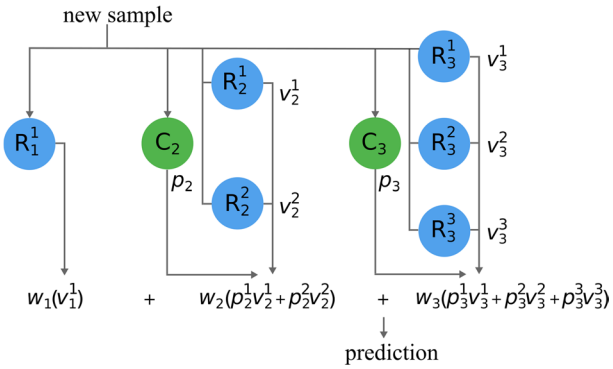

Fig. 1 Representation of the proposed approach when the maximum bin size is 3. Note that the weights (w_1, w_2, w_3) are learned using cross-validation on the learning set using stacking. R_i^j represents the regressor for the jth bin when the bin size is i. p_i^j is the predicted probability by the C_i classifier that a new sample belongs to the jth bin when bin size is i. In the diagram, R_1^1 is the base case regressor and does not have a classifier

size that is not the base case (bin size = 1), and (ii) learning weights to combine the predictions made for each bin size. In the former, there is a classifier and i regressors when bin size is greater than 1. For a new sample, the regressors are used to predict the response while the classifier(s) are used to determine how much we can trust those predictions and weighting them accordingly, see Fig. 1. This differs from a more traditional ensemble learning procedure like stacking in that we do not generate meta-features on which a meta-learner is built, which is then used to aggregate the predictions. This scheme is more closely related to the concept of local classifiers in hierarchical classification (Silla & Freitas, 2011). Furthermore, this can be thought of as a form of dynamic weighting, where instead of having a separate meta-feature generation-aggregation step (Mendes-Moreira et al., 2012), dynamic weighting is implicit in the classifier-regressor aggregation paradigm.

We use a traditional stacking procedure to learn weights to combine the predictions for each bin size (from the base case, where the bin size = 1, to the determined maximum bin size). We generate a meta-feature for each bin size using cross-validation and determine the aggregating weights using a meta-learner. During the testing phase, these weights are applied uniformly to all samples in a test set, making it a static weighting procedure, in contrast to dynamic weighting, where a unique weight is learned for each individual test sample. This approach was chosen to reduce the overall computational complexity of the extended algorithm. However, it is possible that a dynamic weighting paradigm might improve predictive accuracy. We do not prune the meta-feature space, which is one of the main processes in a stacking procedure (Mendes-Moreira et al., 2012). This is because the meta-feature space is not expected to be very large and requiring pruning.

Response discretization is a fundamental part of our algorithm, for which supervised and unsupervised methods have previously been proposed (Dash et al., 2011; Dougherty et al., 1995). Although both supervised and unsupervised approaches are compatible with what we propose, we considered only unsupervised techniques in our evaluation to reduce the computational cost of our experiments. However, one can conjecture that if the supervised methods produce cleaner delineations between the different bins for a given response, it could improve overall predictive accuracy, as the class data on which the classifiers are built would be less noisy. There are other methods which also perform regression by means of classification in an ensemble setting—the work by Ahmad et al. (2018), Halawani et al. (2012) and Gonzalez et al. (2012), for example, are most closely related to what we propose. Ahmad et al. focus on extreme randomized discretization, and use the minimum or maximum of the training data points and bin boundary to estimate the prediction for a new sample, in contrast to our multi-bin and classifier-regressor paradigm. The approach by Gonzalez et al. focus on multi-variate spatio-temporal data and differ from what we propose in two important ways; (a) authors are interested in classifying bands of attributes prior to performing regression, and (b) the final prediction for an unseen sample is obtained by taking the median of the predicted values made by best models which are selected using cross-validation.

An alternative approach to treating imbalanced data is to redress ratio of "rare" and "common" values in a dataset by strategic resampling. Inspired by a similar problem of imbalanced classes in classification, Torgo et al. (2015) proposed two resampling strategies for continuous outcomes in regression analysis. Both methods rely on the notion of *relevance* (Torgo & Ribeiro, 2007), a function mapping output variable domain to [0, 1]. Relevance is inversely proportional to probability density function of the output variable with rarer observations having higher relevance and more common observations having lower relevance. Choosing a relevance threshold allows one to divide the dataset into two subsets:

"common" observations and "rare" observations. Torgo et al. propose two schemes: (i) undersampling of common observations (undersampler) and (ii) undersampling of common observations combined with oversampling of rare observations (SmoteR). Both schemes attempt to create a more balanced dataset allowing for a better representation of values in low probability density areas. Undersampling consists of randomly selecting a subset of normal observations of a pre-specified size. In oversampling, additional "synthetic" rare observations are created by interpolation between existing rare cases and one of its k nearest neighbours. Any regressor can then be applied to a new, re-balanced dataset.

Both, undersampling and SmoteR, are honed towards datasets with rare values concentrated in tail(s) of the distribution. For such distributions, relevance function can be easily calculated and has one or two "relevance bumps"—areas of high relevance, local maxima. However, for more sophisticated distributions (e.g. bimodal), the user has to provide bespoke parameters for construction of the relevance function. This is in contrast to our method, where binning of data is automated via a chosen discretization method. We note that these resampling methods can also be used as part of our proposed framework (see Sect. 6).

3 Methodology

3.1 Algorithm

The original algorithm is described formally in Algorithm 1 and the extension in Algorithm 2. However, we also provide an informal description of the extended algorithm below and a descriptive figure (see Fig. 2). Like the original algorithm, the proposed extension is also split into a build and a prediction phase. In the build phase, the required inputs are a learning set with a continuous response one is interested in modelling, a specified maximum bin size, an unsupervised discretizer, and classification and regression algorithms of one's choosing. The following steps are performed:

1. Discretize the learning set into n bins, where n runs from 1 to the maximum bin size. Note that $n = 1$ is the base case and so no discretization is performed. For example, for a maximum bin size of 3, we end up with three response sets: set 1—base case, set 2—learning set discretized into 2 bins, and set 3—learning set discretized into 3 bins (see Fig. 1).
2. Build a regressor for the base case. For response sets 2 up to the specified maximum bin size, build a single classifier which classifies whether a new sample belongs to a certain bin, and build a regressor for each bin. Therefore, a single classifier and two regressors are built for response set 2, a single classifier and 3 regressors are built for response set 3, and so on.
3. Learn weights that will be used for combining the predictions from the different response sets using the standard stacking procedure. That is, for each response set, generate a meta-feature using the previous two steps and learn combining weights that will be applied to the predictions made for unseen samples. Using these meta-features, also identify the single best performing bin size, which may be the base case.

It is worth noting that depending on the choice of discretizer and response, one can specify a maximum bin size that is too large, and the number of discretized bins fails to reach

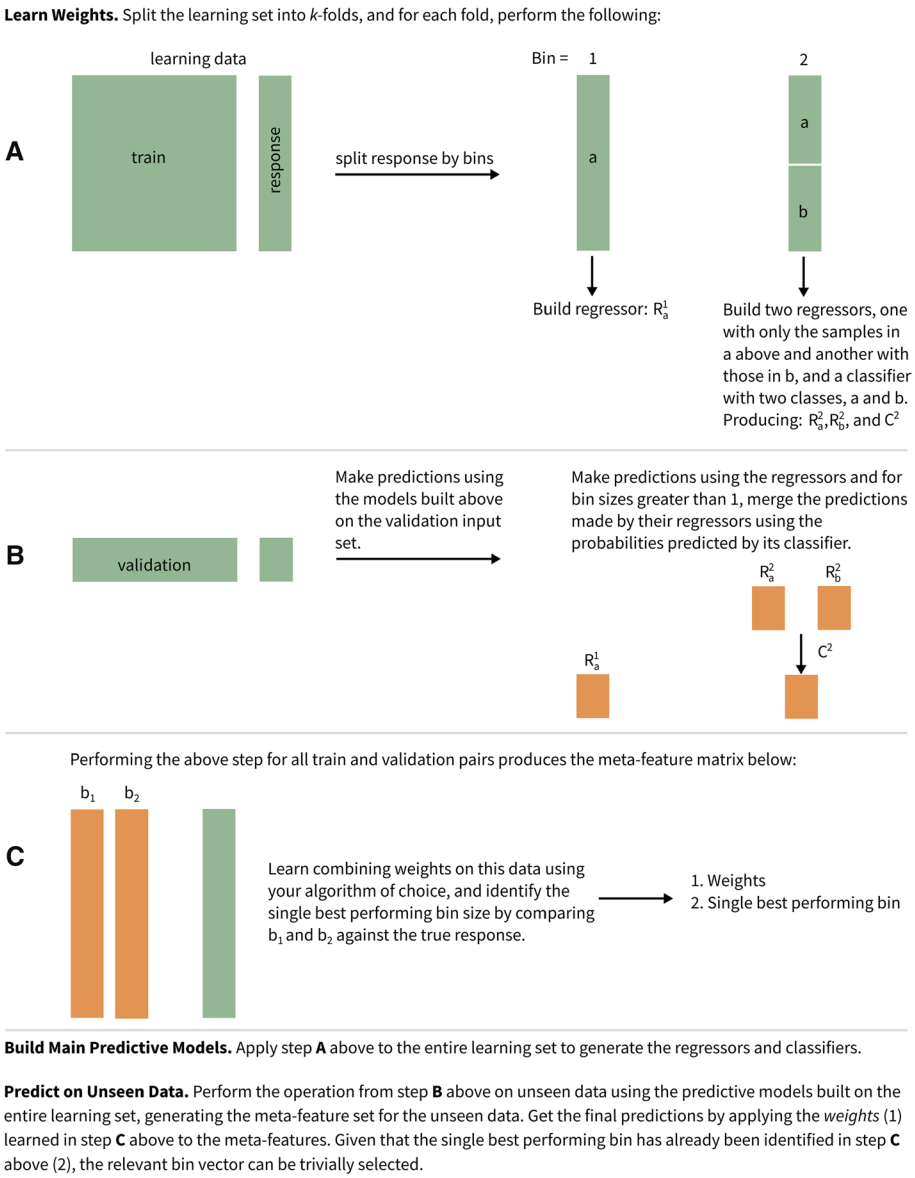

Fig. 2 Data-centric representation of the proposed approach where the max bin size equals 2

the specified size. In this case, the algorithm can be implemented in such a way that it automatically detects the maximum possible bin size before any models are built. This can be achieved by splitting the learning set response into random cross-validation folds, then holding out each fold and attempting to discretize the remaining values at the specified maximum bin size. The minimum discretization size achieved during this cross-validation procedure is chosen as the new maximum. This is how we implemented the algorithm in our evaluation. Given a new sample, the following is performed in the prediction phase:

1. Make predictions for the base case using the base case regressor. Then for response sets 2 up to the maximum bin size, make predictions using the regressors and aggregate them using the paired classifier.
2. After the previous step, one should be left with a number of predictions equal to the maximum bin size. These predictions are then aggregated using the weights from (3) in the building phase. The single best performing bin size can also be identified using the results from (3) in the building phase.

3.2 Considerations

A major consideration is that of practicality from a model building computational expense perspective. Imagine that one has a response they are interested in predicting and let s be the determined maximum bin size. This means that $s-1$ classifiers and $\sum_{l=1}^{s} l$ regressors will be built, making a total of $(s-1)+\sum_{l=1}^{s} l \sim s^2$ models, excluding any additional models built to determine the best hyperparameters for the chosen learning algorithms. Given the k cross-validation folds required to learn the combining weights, the total number of required models built during the learning phase is $k((s-1)+\sum_{l=1}^{s} l)+(s-1)+\sum_{l=1}^{s} l \sim ks^2$ or $\mathcal{O}(s^2)$ models, which is more expensive than the $2s$ or $\mathcal{O}(s)$ models built in the original algorithm. Intuitively, the obvious benefits of this additional computational cost are that (i) one need not know the best performing bin size apriori, and can specify an arbitrarily large value for the maximum bin size, which the algorithm will automatically scale to the most viable, and (ii) between the weighted prediction and the identified best bin, one is guaranteed to never do significantly worse than the base case.

Implicit in this implementation is the assumption that one will use upsampling to resolve class imbalance issues. Empirical evidence from our previous work indicates that this is necessary, where we found that upsampling outperformed all other methods for handling class imbalance in this setting (Orhobor et al., 2020). However, replacing it with one's preferred method is trivial. The only other major decision which is left to the discretion of the user is the choice of discretizer, which can either be done using supervised or unsupervised approaches.

Algorithm 1 Federated Ensemble Learning using Classification. Adapted from Orhobor et al., 2020.

Input: Training set matrix $\mathbf{L} \in \mathbb{R}^{m \times b}$, response vector y, bin size c, and test set matrix $\mathbf{T} \in \mathbb{R}^{n \times b}$

Output: Test set predictions

Training:

Split y into c bins using a discretization technique of choice, producing $\mathcal{L} = (\mathbf{L}_1, \ldots, \mathbf{L}_c)$ and $\mathcal{Y} = (y_1, \ldots, y_c)$

for each bin c in $\mathcal{L}$ and $\mathcal{Y}$ **do**

    Build a regressor R_c using $\mathbf{L}_c$ and y_c

    Build a classifier C_c using $\mathbf{L}_c$ as the positive samples and $\mathcal{L} - \mathbf{L}_c$ as negative samples. *Note: class balancing may be required*

end for

Prediction:

for each c and regressor-classifier pair R_c and C_c **do**

    Predict the response for $\mathbf{T}$ using R_c

    Predict the probability that the samples in $\mathbf{T}$ belong in c using C_c

end for

The process above generates predicted response and probability matrices $\mathbf{R}, \mathbf{P} \in \mathbb{R}^{n \times c}$

$v_n = \sum_{j=1}^{c} p_{n,j}$

Create weight matrix $\mathbf{W} \in \mathbb{R}^{n \times c}$ by dividing all elements in each row i in $\mathbf{P}$ by v_i

Create weighted response matrix $\mathbf{R}^w \in \mathbb{R}^{n \times c}$ by performing the element-wise multiplication of $\mathbf{R}$ and $\mathbf{W}$

The final prediction $T = \sum_{j=1}^{c} r_{n,j}^{w}$

return T

Algorithm 2 Imbalanced Regression using Regressor-Classifier Ensembles.

Input: Learning set matrix $\mathbf{L} \in \mathbb{R}^{m \times d}$, response vector y, s maximum bin size, k cross-validation folds, and test set matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$

Output: Weighted test set predictions and best performing individual bin size predictions

Validate maximum bin size:

1: Validate maximum bin size s. Randomly split y into k folds. Discretize each y_k, where bin size is set to s. For some discretizers and y_k, the maximum number of possible bins is less than s, set s to the minimum of bins from discretizing all y_k.

Learn combining weights:

2: **for** each $1 \ldots k$ folds **do**
3: Split $\mathbf{L}$ into a train ($\mathbf{L}_T$) and validation ($\mathbf{L}_V$) set, where $_V$ represents the samples in k, and $_T$ represents the remaining samples in $\mathbf{L}$
4: Split y_T into i response sets for all $i = 1, \ldots, s$, where $i = 1$ is the base case and no discretization is performed, but for $2 \leq i \leq s$ discretize y_T into i bins.
5: **for** i in $1, \ldots, s$ response sets **do**
6: **if** $i = 1$ (base case) **then**
7: Build a regressor for the base case R_i using $\mathbf{L}_T$
8: Make predictions for $\mathbf{L}_V$ using R_i, producing a single prediction vector r_i
9: **else**
10: Build a classifier C_i using $\mathbf{L}_T$, where each bin b is a class, upsampling the minority class when necessary
11: Build a regressor for each bin b, R_i^b, where $b = 1, \ldots, i$, using the appropriate samples in $\mathbf{L}_T$
12: Make predictions for $\mathbf{L}_V$ using all R_i^b, producing b prediction vectors r_i^b
13: Merge r_i^b into a single prediction vector by averaging the predictions using the probabilities predicted by C_i for $\mathbf{L}_V$, producing r_i
14: **end if**
15: **end for**
Steps 5-15 creates a validation meta-feature matrix $\mathbf{V}_k \in \mathbb{R}^{v \times s}$, where v is the number of samples in the validation set and s is the number of response sets (r_i)
16: **end for**
17: Combine all $\mathbf{V}_k$ into a single meta-feature matrix $\mathbf{M} \in \mathbb{R}^{m \times s}$, where m is the number of samples in the learning set and s is the number of response sets. Learn combining weights w on $\mathbf{M}$, and identify the best performing individual response set.

Main model building:

18: Split y into c bins using a discretization technique of choice, producing $\mathcal{L} = (\mathbf{L}_1, \ldots, \mathbf{L}_c)$ and $\mathcal{Y} = (y_1, \ldots, y_c)$
19: **for** each bin c in $\mathcal{L}$ and $\mathcal{Y}$ **do**
20: Build a regressor R_c using $\mathbf{L}_c$ and y_c
21: Build a classifier C_c using $\mathbf{L}_c$ as the positive samples and $\mathcal{L} - \mathbf{L}_c$ as negative samples. *Note: class balancing may be required*
22: **end for**

Prediction:

23: **for** i in $1, \ldots, s$ **do**
24: **if** $i = 1$ (base case) **then**
25: Predict the response r_i for $\mathbf{T}$ using $R_{i=1}$
26: **else**
27: Predict the response for $\mathbf{T}$ using all R_i^b, where $b = 1, \ldots, i$. This produces a response matrix $\mathbf{R} \in \mathbb{R}^{n \times i}$
28: Predict the probability that the samples in $\mathbf{T}$ belong the b bins. This should produce a classification matrix $\mathbf{C} \in \mathbb{R}^{n \times i}$
29: The final response $r_i = \sum_{j=1}^{i} \mathbf{Y}_{nj}$, where $\mathbf{Y} = (\mathbf{R})_{nj}(\mathbf{C})_{nj}$
30: **end if**
31: **end for**
The process above generates a predicted response matrix $\mathbf{P} \in \mathbb{R}^{n \times s}$
32: The final weighted prediction W is gotten by applying the learned weights w from step 17 to $\mathbf{P}$ above. The prediction S, which should be treated as the best performing individual response set is trivially identified by comparing each meta-feature which maps to a particular bin size in $\mathbf{M}$ on step 17 against the true values in the learning set.
33: **return** W and S

4 Evaluation setup

4.1 Datasets

We evaluated the proposed algorithm and the two resampling methods using 119 responses from a variety of datasets. Specifically, we considered the 20 genes from the LINCS Phase II dataset (Koleti & Terryn, 2017) which we used in the evaluation of the original algorithm (Algorithm 1), 46 traits from the yeast dataset used by Grinberg et al. (2020), 24 drug targets from quantitative structure-activity relationship problems (Olier et al., 2018), 14 responses from the OpenML AutoML Regression Benchmark (Bischl et al., 2017), and 15 responses from work by Branco et al. (2019). We refer to these dataset *collections* as gene expression, yeast, QSAR, OpenML, and Branco, respectively, during the discussion of our evaluation in Sect. 5. The exact versions of these datasets used in our experiments are available for download from http://dx.doi.org/10.17632/mpvwnhv4vb.2. We used a 70–30% learning and test set split in our evaluation. See Tables in "Appendix A" for the distribution of imbalance in the considered responses as described by Branco et al. (2019).

4.2 Learning setup

We considered the unsupervised frequency interval and clustering (k-means) discretizers (Dougherty et al., 1995). The specified maximum bin size was 10, 5-fold cross-validation was used in the internal stacking procedure to learn weights and identify the best performing individual bin size on the *learning set*. We used convex linear regression as our weight learning algorithm, which is multiple linear regression with the constraint that its coefficients are non-negative and must sum to 1 (Breiman, 1996). For the classifiers, we used random forests (RF) (Breiman, 2001), naive bayes (NB) (Rish, 2001), decision trees (DT) (Quinlan, 1986), and k-nearest neighbours (KNN) (Altman, 1992). For the regressors, we used the least absolute shrinkage and selection operator (LS) (Tibshirani, 1996), ridge regression (RR) (Hoerl & Kennard, 1970), RF, and eXtreme gradient boosting (XGB) (Chen & Guestrin, 2016). The RF models were built with 1000 trees and a third of the number of attributes was used as the number of splitting variables considered at each node, the defaults were used for everything else. The KNN models were tuned for the k parameter (the number of neighbours) over a range of values: (1, 3, 5, 7, 11, 13, 15). The regularization parameter for both LS and RR was tuned using 10-fold internal cross-validation. Lastly, the learning rate of the XGB models were tuned with (0.001, 0.01, 0.1, 0.2, 0.3) and the number of rounds set to 1500. Default values were used for all other parameters. Note that, out of the four regression models, only RF is capable of handling non-binary categorical input variables. Hence, LS, RR and XGB were not applied to datasets which included such inputs. Predictive performance was measured as the coefficient of determination (R^2). All experiments were performed in R (Ihaka & Gentleman, 1996), and the code used in our experiments is available at https://github.com/oghenejokpeme/EFERUC. All our results are available for independent analysis at http://dx.doi.org/10.17632/mpvwnhv4vb.2.

4.3 SmoteR and undersampler

We compared our proposed method with two resampling approaches from Torgo et al. (2015), SmoteR and undersampler. For SmoteR, we used a relevance threshold of 0.7, $k = 5$ and all combinations of undersampling rate of $(50, 100, 200, 300)\%$ and oversampling rate of $(200, 500)\%$. For undersampler, we used a relevance threshold of 0.7 and resampling rates of $(50, 100, 200, 300)\%$. These combination of parameters were used in the original paper (Torgo et al., 2015). The results from the best performing parameters are reported. We applied the four regression methods (LS, RR, RF and XGB), which were tuned as detailed above, to each resampled dataset and reported the R^2 on a test set for the best combination of resampling parameters. For majority of outputs, the relevance function was generated automatically (the `phi` function in the `uba` package in `R`) with either one or two relevance bumps corresponding to tail(s) of the distribution. However, some outputs required bespoke relevance functions with areas of importance identified through inspection of the output histograms. We note that the undersampling algorithm and SmoteR failed in the instances when no extreme values were identified by the relevance function and in some rare cases where a dataset contained discrete or categorical input variables with very low-frequency values.

5 Results

5.1 Base regressor performance

We found that the overall best performing regressor in the base case for the frequency and clustering based discretization approaches was XGB, with RF a close second (see Tables 1 and 2). One might ask why we have separated the base case results by discretizer given that the results should be the same either way. This is because we only included base case results where the proposed approach could be successfully executed for at least one regressor-classifier pair given a particular discretizer. This is important because the learning process can fail depending on how a response is discretized. Failure in this sense can be caused by either the inability to discretize a response beyond a bin size of 1, or the discretization could be performed, but all the values in one bin were

Table 1 Base case mean predictive performance (R^2) for the regressors we considered in our evaluation where the discretization approach is *frequency* for the data collections we considered

Dataset	LS	RR	RF	XGB
Branco (3)	0.793 ± 0.12	0.794 ± 0.12	0.923 ± 0.05	$\mathbf{0.943 \pm 0.04}$
Gene expression (20)	0.069 ± 0.06	0.074 ± 0.06	0.086 ± 0.1	$\mathbf{0.088 \pm 0.09}$
OpenML (8)	0.225 ± 0.23	0.232 ± 0.22	$\mathbf{0.434 \pm 0.27}$	0.357 ± 0.27
QSAR (24)	0.570 ± 0.06	0.582 ± 0.05	$\mathbf{0.676 \pm 0.06}$	0.660 ± 0.06
Yeast (46)	$\mathbf{0.439 \pm 0.15}$	0.392 ± 0.13	0.380 ± 0.15	0.419 ± 0.15

Note that for each data collection, we only included results from responses where (1) performance values are available for all learners, and (2) each regressor could be successfully paired with at least one other classifier without failure. The number of experiments for each data subset is given and the best performing regressor is in boldface

Table 2 Base case mean predictive performance (R^2) for the regressors we considered in our evaluation where the discretization approach is *clustering* for the data collections we considered

Dataset	LS	RR	RF	XGB
Branco (3)	0.793 ± 0.12	0.794 ± 0.12	0.923 ± 0.05	**0.943 ± 0.04**
Gene expression (19)	0.059 ± 0.05	0.064 ± 0.05	0.072 ± 0.08	**0.074 ± 0.06**
OpenML (14)	0.395 ± 0.35	0.398 ± 0.34	**0.552 ± 0.34**	0.520 ± 0.37
QSAR (23)	0.571 ± 0.06	0.582 ± 0.06	**0.676 ± 0.06**	0.660 ± 0.06
Yeast (46)	**0.439 ± 0.15**	0.392 ± 0.13	0.380 ± 0.15	0.419 ± 0.15

Note that for each data collection, we only included results from responses where (1) performance values are available for all learners, and (2) each regressor could be successfully paired with at least one other classifier without failure. The number of experiments for each data subset is given and the best performing regressor is in boldface

the same, causing some learners to fail. It is also worth noting that we did not perform experiments for base regressors which only support numeric input variables but the input dataset under consideration has nominal variables with more than two classes. For example, we did not build a LS regressor for fuelCons dataset in the Branco collection (see Table 11 in "Appendix A"). Paired t-tests showed that there is significant difference in performance between the regressor pairs in the base case for some dataset collections but not for others (see Tables 3 and 4).

Table 3 *P*-values from paired t-tests for each unique regressor pair for the data subsets (including the number of experiments) we considered when the discretization approach is *frequency*

Pair	Branco (3)	Gene expression (20)	OpenML (8)	QSAR (24)	Yeast (46)
LS-RR	0.225	6.010e-06	0.327	2.258e-09	3.832e-06
LS-RF	0.233	0.058	0.024	1.981e-15	3.619e-06
LS-XGB	0.172	0.004	0.023	3.120e-15	0.028
RR-RF	0.238	0.152	0.030	9.416e-15	0.448
RR-XGB	0.175	0.022	0.038	1.594e-14	0.050
RF-XGB	0.130	0.907	0.314	1.374e-06	5.248e-09

Table 4 *P*-values from paired t-tests for each unique regressor pair for the data subsets (including the number of experiments) we considered when the discretization approach is *clustering*

Pair	Branco (3)	Gene expression (19)	OpenML (14)	QSAR (23)	Yeast (46)
LS-RR	0.225	1.639e-05	0.433	7.883e-09	3.832e-06
LS-RF	0.233	0.111	0.020	1.306e-14	3.619e-06
LS-XGB	0.172	0.005	0.013	2.012e-14	0.028
RR-RF	0.238	0.276	0.022	5.792e-14	0.448
RR-XGB	0.175	0.033	0.017	9.520e-14	0.050
RF-XGB	0.130	0.913	0.481	3.452e-06	5.248e-09

5.2 Classifier performance

We observed that the best performing classifier, when paired with a particular regressor, is largely dependent on the discretization approach and varies by dataset collection. For example, RF is the best performing classifier on the Branco dataset collection for all regressors when frequency is the discretization approach. This is in contrast to when clustering is used for discretization. In this case, KNN is the best classifier for LS and RR, RF is the best classifier for the RF regressor, and NB and DT perform equally well when paired with XGB (see Tables 5 and 6). Given these results, we argue that for a new unseen problem, one would need to also select the best performing classifier that should be paired with a given regressor. This can be done using any standard model selection approach.

Table 5 Mean predictive performance (R^2) of the best performing of the best individual bin size and weighted cases for each regressor-classifier pair we considered when the discretization approach is *frequency*

	n	NB	RF	DT	KNN
Branco					
LS	3	0.819 ± 0.10	**0.898 ± 0.09**	0.830 ± 0.13	0.857 ± 0.10
RR	3	0.822 ± 0.10	**0.895 ± 0.10**	0.834 ± 0.14	0.860 ± 0.09
RF	12	0.724 ± 0.35	**0.726 ± 0.35**	0.231 ± 0.42	0.232 ± 0.42
XGB	3	0.943 ± 0.04	**0.943 ± 0.03**	0.943 ± 0.04	**0.943 ± 0.03**
Gene expression					
LS	20	0.069 ± 0.06	0.097 ± 0.07	0.072 ± 0.06	**0.098 ± 0.08**
RR	20	0.075 ± 0.06	0.097 ± 0.07	0.076 ± 0.07	**0.099 ± 0.07**
RF	20	0.090 ± 0.09	0.110 ± 0.09	**0.121 ± 0.08**	0.091 ± 0.09
XGB	20	0.092 ± 0.09	**0.112 ± 0.08**	0.094 ± 0.09	0.111 ± 0.09
OpenML					
LS	8	0.233 ± 0.23	**0.373 ± 0.22**	0.267 ± 0.22	0.247 ± 0.24
RR	8	0.237 ± 0.22	**0.348 ± 0.25**	0.260 ± 0.22	0.251 ± 0.23
RF	12	**0.574 ± 0.30**	0.510 ± 0.33	0.291 ± 0.30	0.290 ± 0.30
XGB	8	0.360 ± 0.27	**0.411 ± 0.24**	0.365 ± 0.26	0.360 ± 0.27
QSAR					
LS	24	0.572 ± 0.06	0.639 ± 0.06	0.574 ± 0.06	**0.642 ± 0.06**
RR	24	0.583 ± 0.05	0.641 ± 0.06	0.583 ± 0.06	**0.645 ± 0.06**
RF	24	0.676 ± 0.06	0.677 ± 0.06	0.676 ± 0.06	**0.679 ± 0.06**
XGB	24	0.660 ± 0.06	0.667 ± 0.06	0.660 ± 0.06	**0.672 ± 0.05**
Yeast					
LS	46	**0.445 ± 0.15**	0.442 ± 0.15	0.445 ± 0.16	0.440 ± 0.15
RR	46	0.401 ± 0.14	0.404 ± 0.14	**0.426 ± 0.15**	0.393 ± 0.13
RF	46	**0.398 ± 0.15**	0.381 ± 0.15	0.382 ± 0.15	0.382 ± 0.15
XGB	46	**0.425 ± 0.15**	0.420 ± 0.15	0.421 ± 0.15	0.419 ± 0.15

Individual performance values (not entire entries) of less than zero and missing values (cases where a classifier building process failed) were replaced with 0s. For each data collection, the number of experiments for each regressor *n* is given, and the best performing classifier is in boldface

Table 6 Mean predictive performance (R^2) of the best performing of the best individual bin size and weighted cases for each regressor-classifier pair we considered when the discretization approach is *clustering*

	n	NB	RF	DT	KNN
Branco					
LS	3	0.550 ± 0.49	0.604 ± 0.41	0.694 ± 0.44	**0.892 ± 0.10**
RR	3	0.551 ± 0.49	0.612 ± 0.40	0.695 ± 0.45	**0.891 ± 0.10**
RF	9	0.900 ± 0.13	**0.903 ± 0.13**	0.308 ± 0.46	0.309 ± 0.46
XGB	3	**0.943 ± 0.04**	0.942 ± 0.03	**0.943 ± 0.04**	0.942 ± 0.03
Gene expression					
LS	19	0.059 ± 0.05	0.069 ± 0.05	0.060 ± 0.05	**0.081 ± 0.05**
RR	19	0.064 ± 0.05	0.071 ± 0.05	0.065 ± 0.05	**0.084 ± 0.05**
RF	19	0.074 ± 0.07	0.077 ± 0.07	**0.086 ± 0.07**	0.074 ± 0.07
XGB	19	0.078 ± 0.06	0.086 ± 0.06	0.079 ± 0.06	**0.091 ± 0.06**
OpenML					
LS	14	0.408 ± 0.35	**0.552 ± 0.35**	0.447 ± 0.36	0.439 ± 0.38
RR	14	0.411 ± 0.34	0.536 ± 0.37	**0.444 ± 0.36**	0.441 ± 0.37
RF	14	0.552 ± 0.34	**0.557 ± 0.33**	0.550 ± 0.34	0.546 ± 0.33
XGB	14	0.527 ± 0.37	**0.570 ± 0.36**	0.533 ± 0.36	0.523 ± 0.37
QSAR					
LS	23	0.571 ± 0.06	**0.649 ± 0.06**	0.573 ± 0.06	0.637 ± 0.06
RR	23	0.583 ± 0.06	**0.651 ± 0.06**	0.584 ± 0.06	0.640 ± 0.06
RF	23	0.676 ± 0.06	0.677 ± 0.06	0.676 ± 0.06	**0.678 ± 0.06**
XGB	23	0.660 ± 0.06	0.669 ± 0.06	0.660 ± 0.06	**0.670 ± 0.06**
Yeast					
LS	46	0.444 ± 0.15	0.443 ± 0.15	**0.447 ± 0.16**	0.440 ± 0.15
RR	46	0.399 ± 0.14	0.408 ± 0.14	**0.426 ± 0.16**	0.393 ± 0.13
RF	46	0.396 ± 0.15	0.381 ± 0.15	**0.382 ± 0.15**	0.381 ± 0.15
XGB	46	**0.426 ± 0.15**	0.420 ± 0.15	0.421 ± 0.15	0.419 ± 0.15

Performance values less than zero and missing values (cases where the classifier building process failed) were replaced with 0s. For each data subset, the number of experiments for each regressor *n* is given, and the best performing classifier is in boldface

5.3 Discretizer performance

Our results suggests that the effect of the chosen discretizer is dependent on the response and the regressor-classifier pair. We only see a statistically significant difference in performance between the frequency and clustering discretizers in 13 of the 80 regressor-classifier pairs across the dataset collections we considered. However, the choice of discretizer might be worth optimising given a new problem, as the analysis for the gene expression dataset collection indicates significant effects on performance (see Table 7).

Table 7 The percentage of regressor-classifier pairs for which the considered discretization approaches (*frequency*/*clustering*) strictly outperforms the other

	n	NB	RF	DT	KNN
Branco					
LS	3	66.7/33.3	66.7/33.3	33.3/66.7	0/66.7
RR	3	66.7/33.3	66.7/33.3	33.3/66.7	0/66.7
RF	9	11.1/0	11.1/33.3	0/0	11.1/11.1
XGB	3	0/0	66.7/0	0/0	33.3/66.7
Gene expression					
LS	19	26.3/10.5	100/0*	78.9/5.3*	78.9/21.1
RR	19	15.8/0	100/0*	57.9/15.8	68.4/15.8
RF	19	52.6/10.5*	100/0*	100/0*	26.3/5.3
XGB	19	42.1/26.3	94.7/0*	42.1/36.8	68.4/21.1*
OpenML					
LS	8	37.5/25	50/37.5	50/25	50/37.5
RR	8	37.5/37.5	37.5/25	37.5/37.5	37.5/37.5
RF	11	9.1/9.1	27.3/18.2	9.1/27.3	27.3/36.4
XGB	8	37.5/12.5	50/12.5	37.5/12.5	25/37.5
QSAR					
LS	23	39.1/17.4	21.7/73.9*	52.2/13	65.2/26.1*
RR	23	17.4/13	8.7/87*	30.4/34.8	69.6/30.4*
RF	23	8.7/0	8.7/21.7	0/0	30.4/26.1
XGB	23	4.3/0	34.8/39.1	0/4.3	56.5/26.1
Yeast					
LS	46	43.5/21.7	17.4/13	23.9/39.1	13/13
RR	46	43.5/19.6	13/45.7*	39.1/34.8	15.2/21.7
RF	46	54.3/26.1	10.9/2.2	13/17.4	17.4/13
XGB	46	32.6/32.6	2.2/0	10.9/19.6	2.2/4.3

Note that the value for each regressor-classifier pair in this comparison is the best performer of the best bin or weighted case. The number of experiments (n) for each pair is given, and an asterisk (*) is given where there is a statistically significant difference in performance between the discretization approaches on a paired t-test

5.4 Algorithm performance

We compared the base regressor performance to the best performing regressor-classifier pair for the frequency and clustering discretizers. Note that the values for the regressor-classifier pairs here are the best performing of either the best performing single bin or the weighted case. For example, assume a predicted response and LS as the base regressor. We compared the base LS performance value to the best performing pair out of LS-NB, LS-RF, LS-DT, and LS-KNN, where the value for each of these pairs is the best performer out of the best performing single bin and the weighted case. There were a total of 853 performance data points. Our approach performed as well as the base case on 134 and strictly outperformed it on 709. The base case never outperforms our proposed approach. For the cases in which our proposed approach outperformed the base case, the average increase in predictive performance (R^2) is

Table 8 *P*-values from paired t-test significance testing of the difference in performance between the regressor base case and the best performing regressor-classifier pair (best of best bin and weighted). *n* is the number of samples used in the significance tests

	p-value	*n*	Mean R^2 increase
Frequency			
LS	$1.086e^{-12}$	92	.05 ± .06
RR	$2.041e^{-12}$	93	.05 ± .06
RF	$1.093e^{-15}$	86	.02 ± .02
XGB	$4.482e^{-10}$	78	.02 ± .03
Clustering			
LS	$6.146e^{-08}$	91	.06 ± .09
RR	$1.056e^{-08}$	99	.06 ± .09
RF	$6.662e^{-12}$	85	.01 ± .02
XGB	0.002	85	.02 ± .06

Last column is mean increase in R^2 of the best performing regressor-classifier pair compared to the base case (± standard deviation)

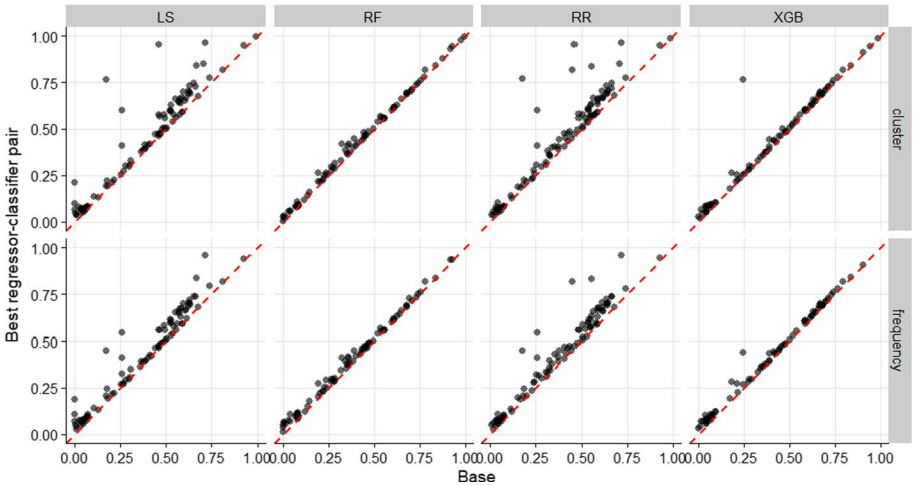

Fig. 3 Comparison of performance (R^2) of the base case and the best performing regressor-classifier pair for each regressor and discretization approach, across all datasets. Dashed red line is the $x = y$ line

0.04 ± 0.06 (an average increase of performance of ~35%). We performed significance testing using paired t-tests for each individual regressor given the two discretizers we considered. This analysis showed that the difference in performance is statistically significant for the four regressors with a significance level of 0.01 (see Table 8 and Fig. 3). The most marked improvement in performance is observed for the two linear methods, LS and RR, for both discretization approaches, whilst performance of RF and XGB is very similar for base and best regressor-classifier pair. These results suggest that the proposed approach (a) is capable of dealing with the problem of imbalanced distributions in a target response, and (b) is highly unlikely to perform worse than the base case.

5.5 Comparison to resampling approaches

We compared the best performing resamplers (SmoteR and undersampler) to the best performing regressor-classifier pair for the frequency and clustering discretizers. Also note that, as in the previous section, the values for the regressor-classifier pairs here are the best performing of either the best performing single bin or the weighted case. We observed that for all combinations of the resamplers, discretizers, and dataset collections, our approach outperforms the resampling methods (see Table 9 and Fig. 4). Paired t-tests show that these differences in performance are statistically significant with a significance level of 0.01 (see Table 10).

6 Discussion and conclusion

Although we have demonstrated the utility of the proposed algorithm, we conjecture that modifications to three main components could further improve the predictive accuracy of the algorithm's output at some additional computational cost. These components

Table 9 Comparison of the resampling approaches (SmoteR and undersampler) to the discretizers (frequency and clustering) used in the evaluation of our proposed methods across the considered collections of datasets (B—Branco, G—Gene expression, O—OpenML, Q—QSAR, Yeast - Y)

Resampler	Discretizer	Dataset	*n*	*eq*	*lt*	*lt_avg*	*gt*	*gt_avg*
SmoteR	Frequency	B	21	1	7	.01 ± .01	**13**	.08 ± .12
SmoteR	Frequency	G	80	0	0	–	**80**	.08 ± .03
SmoteR	Frequency	O	27	3	6	.02 ± .03	**18**	.15 ± .12
SmoteR	Frequency	Q	84	0	0	–	**84**	.12 ± .08
SmoteR	Frequency	Y	180	0	14	.03 ± .03	**166**	.08 ± .07
SmoteR	Clustering	B	18	2	5	.01 ± .01	**11**	.08 ± .13
SmoteR	Clustering	G	76	0	0	–	**76**	.07 ± .03
SmoteR	Clustering	O	35	7	4	.03 ± .04	**24**	.20 ± .22
SmoteR	Clustering	Q	80	0	0	–	**80**	.12 ± .08
SmoteR	Clustering	Y	180	1	14	.03 ± .03	**165**	.08 ± .07
Undersampler	Frequency	B	21	0	2	.00 ± .00	**19**	.08 ± .11
Undersampler	Frequency	G	80	0	0	–	**80**	.07 ± .03
Undersampler	Frequency	O	31	1	1	.01 ± –	**29**	.13 ± .11
Undersampler	Frequency	Q	84	0	0	–	**84**	.22 ± .12
Undersampler	Frequency	Y	180	0	2	.01 ± .00	**178**	.10 ± .08
Undersampler	Clustering	B	18	0	0	–	**18**	.08 ± .11
Undersampler	Clustering	G	76	0	0	–	**76**	.06 ± .03
Undersampler	Clustering	O	40	5	1	.01 ± –	**34**	.18 ± .20
Undersampler	Clustering	Q	80	0	0	–	**80**	.23 ± .13
Undersampler	Clustering	Y	180	1	3	.01 ± .00	**176**	.10 ± .08

n is the number of samples that were compared, *eq* is the number of times a resampling and discretization approach perform equally well, *lt* is the number of times a resampling approach outperforms a discretization approach, *lt_avg* is the average performance increase (R^2) and standard deviation of the resampling approach over a discretization approach, and *gt_avg* is the average performance increase (R^2) and standard deviation of a discretization approach over a resampling approach. The overall best performing approach for each comparison (*lt*) or (*gt*) is in boldface

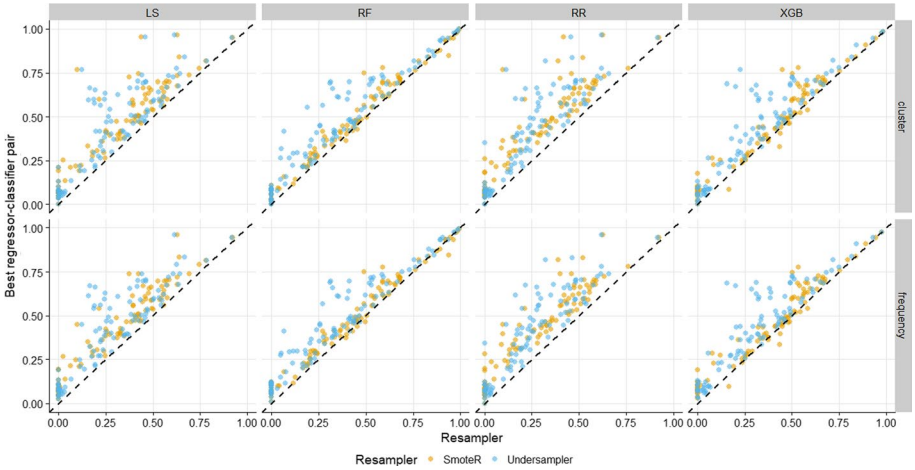

Fig. 4 Comparison of performance (R^2) of the two resampling methods and the best performing regressor-classifier pair for each regressor and discretization approach, across all datasets. Dashed black line is the $x = y$ line

Table 10 *P*-values from paired t-test significance testing of the difference in performance between the resampling methods and the best performing regressor-classifier pair (best of best bin and weighted) for each discretizer across the considered datasets on a per regressor basis

	p-value	*n*
Frequency - SmoteR		
LS	$1.827e^{-26}$	95
RR	$1.255e^{-28}$	95
RF	$5.343e^{-17}$	107
XGB	$2.264e^{-12}$	95
Frequency - Undersampler		
LS	$3.838e^{-20}$	96
RR	$2.259e^{-23}$	96
RF	$7.154e^{-21}$	108
XGB	$1.018e^{-16}$	96
Clustering - SmoteR		
LS	$3.484e^{-19}$	95
RR	$1.321e^{-22}$	95
RF	$7.209e^{-15}$	104
XGB	$4.662e^{-11}$	95
Clustering - Undersampler		
LS	$2.215e^{-17}$	96
RR	$2.053e^{-20}$	96
RF	$5.937e^{-18}$	106
XGB	$2.610e^{-14}$	96

n is the number of experiments used in the significance tests

are the method by which class imbalance is handled, the type of discretizer, and how weighting is performed in the internal stacking procedure. In our evaluation, we used simple upsampling to deal with class imbalance, however, we expect that a more sophisticated but more computationally expensive approach like SMOTE (Chawla et al., 2002) would perform better. The reasoning behind this is that building the classifiers with higher quality samples will lead to improved classification accuracy of the models, which will in turn improve the classification accuracy of the overall system. For the discretizers, we only considered unsupervised discretizers because of their negligible computational cost, however, we also expect overall performance improvement of the proposed algorithm with supervised discretizers. We think that supervised discretizers would produce cleaner delineations between the different bins, thus reducing the amount of noise between bin regions, improving the predictive accuracy of the classifiers. One might also choose to use a bespoke discretization approach that is tailored to a given problem using domain knowledge, which may outperform the mainstream supervised and unsupervised discretization techniques. Lastly, rather than the static weighting scheme we used in aggregating the predictions from the different bin sizes, it is possible that a dynamic weighting approach might perform better. One might argue that depending on their choice for the internal components (discretizer, class balancing, regressor, classifier, etc), the computational cost (finance and time) of the proposed approach might be limiting. However, we argue that this is not the case, given that compute costs are cheap and are steadily declining, and certain aspects of the algorithm can be parallelised, which should dramatically reduce computation time.

In summary, we have presented an extension to the federated ensemble regression using classification algorithm. Our evaluation shows that it is highly unlikely to perform worse than the base case, and it outperforms state-of-the-art resampling approaches intended to address distribution imbalance in target responses for regression problems.

Appendix A: Dataset details

The fraction of rare values was calculated with a threshold of 0.7 using the formulation given in Branco et al. (2019) (Tables 11, 12, 13, 14 and 15).

Table 11 Dataset description for the gene expression targets

Target	Samples	Features	Rare fraction (%)
AKT1	10000	1154	13.87
APOE	10000	1154	17.31
BRCA1	10000	1154	16.22
CDK4	10000	1154	13.35
EGF	10000	1154	17.04
EGFR	10000	1154	12.3
KIT	10000	1154	11.06
PAX8	10000	1154	12.8
TGFBR2	10000	1154	21.69
TP53	10000	1154	14.46
STK10	10000	1154	15.97
CDH3	10000	1154	15.3
IGF1R	10000	1154	13.31
TNFRSF21	10000	1154	17.2
RAD51C	10000	1154	16.69
CFLAR	10000	1154	14.12
TERT	10000	1154	12.37
FGFR2	10000	1154	12.94
LYN	10000	1154	14.32
PTK2	10000	1154	16.02

Table 12 Dataset description for the OpenML datasets

Dataset	Samples	Features	Rare fraction (%)
house sales	21613	21	10.13
abalone	4177	8	21.12
brazillian houses	10692	12	13.58
santander transactions	4459	4991	14.62
mercedes benz	4209	376	4.06
mip	1090	146	22.48
house16h	22784	16	14.18
quake	2178	3	9.78
socmob	1156	5	18.86
space	3107	6	8.43
topo	8885	266	19.08
yprop	8885	251	19.08
wine quality	6497	11	23.44
elevators	16599	18	15.53
pol	15000	48	17.69

Table 13 Dataset description for the Branco datasets

Dataset	Samples	Features	Rare fraction (%)
a1	198	11	16.16
a2	198	11	13.13
a3	198	11	17.17
a4	198	11	18.69
a6	198	11	17.68
a7	198	11	14.65
Abalone	4177	8	21.12
Accel	1732	14	6.12
availPwr	1802	15	10.93
bank8FM	4499	8	8.54
boston	506	13	14.62
cpuSm	8192	12	9.81
fuelCons	1764	37	12.76
heat	7400	11	10.7
maxTorque	1802	32	9.05

Table 14 Dataset description for the QSAR datasets

Target	Samples	Features	Rare fraction (%)
CHEMBL1907610	3083	1024	4.67
CHEMBL203	5012	1024	6.17
CHEMBL204	5742	1024	17.29
CHEMBL205	3664	1024	12.96
CHEMBL214	3356	1024	3.52
CHEMBL226	3490	1024	6.05
CHEMBL228	4156	1024	2.55
CHEMBL233	4089	1024	18.34
CHEMBL234	3133	1024	42.36
CHEMBL251	4081	1024	5.71
CHEMBL253	4332	1024	2.29
CHEMBL256	3438	1024	5.47
CHEMBL260	3889	1024	33.81
CHEMBL261	3019	1024	4.24
CHEMBL262	3058	1024	5.36
CHEMBL264	3134	1024	6.51
CHEMBL267	3650	1024	7.1
CHEMBL284	3429	1024	13.09
CHEMBL313	3527	1024	3.09
CHEMBL332	3459	1024	5.84
CHEMBL333	3617	1024	25.38
CHEMBL339	4378	1024	7.38
CHEMBL340	3649	1024	10.69
CHEMBL344	3048	1024	1.71

Table 15 Dataset description for the yeast dataset

Target	Samples	Features	Rare fraction (%)
Cadmium_Chloride	799	11623	13.52
Caffeine	1004	11623	10.46
Calcium_Chloride	949	11623	10.12
Cisplatin	990	11623	8.18
Cobalt_Chloride	1007	11623	4.47
Congo_red	979	11623	27.17
Copper	972	11623	14.4
Cycloheximide	1004	11623	11.16
Diamide	1003	11623	6.88
E6_Berbamine	1005	11623	10.85
Ethanol	967	11623	10.34
Formamide	999	11623	12.71
Galactose	962	11623	11.43
Hydrogen_Peroxide	769	11623	11.57
Hydroquinone	960	11623	20.1
Hydroxyurea	944	11623	12.29
Indoleacetic_Acid	994	11623	15.79
Lactate	973	11623	8.84
Lactose	1004	11623	11.25
Lithium_Chloride	1002	11623	4.79
Magnesium_Chloride	964	11623	8.51
Magnesium_Sulfate	1006	11623	18.39
Maltose	1005	11623	13.23
Mannose	957	11623	11.91
Menadione	1005	11623	5.17
Neomycin	1005	11623	8.96
Paraquat	1004	11623	6.37
Raffinose	662	11623	10.27
SDS	873	11623	7.79
Sorbitol	599	11623	9.02
Trehalose	1003	11623	7.78
Tunicamycin	1006	11623	23.56
x4.Hydroxybenzaldehyde	964	11623	6.85
x4NQO	1005	11623	21.39
x5.Fluorocytosine	957	11623	12.12
x5.Fluorouracil	992	11623	11.69
x6.Azauracil	1001	11623	8.99
Xylose	1003	11623	7.28
YNB	1006	11623	10.93
YNB.ph3	979	11623	3.98
YNB.ph8	887	11623	8.57
YPD	1006	11623	11.63
YPD.15C	1005	11623	23.18
YPD.37C	1003	11623	9.77
YPD.4C	806	11623	8.06

Table 15 (continued)

Target	Samples	Features	Rare fraction (%)
Zeocin	957	11623	48.69

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) UK through the ACTION on cancer Grant (EP/R022925/1, EP/R022941/1). The computational resources were provided by the Swedish National Infrastructure for Computing (SNIC) at the Chalmers University of Technology partially funded by the Swedish Research Council through Grant Agreement No. 2018-05973. Prof. King acknowledges the support of the Knut and Alice Wallenberg Foundation Wallenberg Autonomous Systems and Software Program (WASP). N. F. Grinberg would like to acknowledge funding from the Wellcome Trust (WT107881) and the MRC (MC_UU_00002/4).

Author Contributions O.I.O., N.F.G., L.N.S., and R.D.K. designed research; O.I.O. and N.F.G performed the experiments; O.I.O., N.F.G., and R.D.K. analyzed data; and O.I.O. and N.F.G wrote the paper with input from co-authors.

Funding This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) UK through the ACTION on cancer Grant (EP/R022925/1, EP/R022941/1).

Availability of data and material All data used in this work is publicly available as reported in Sect. 4.

Code availability All code used in this work is publicly available as reported in Sect. 4.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Ahmad, A., Khan, S. S., & Kumar, A. (2018). Learning regression problems by using classifiers. *Journal of Intelligent & Fuzzy Systems, 35*(1), 945–955.

Ali, A., Shamsuddin, S. M., Ralescu, A. L., et al. (2015). Classification with class imbalance problem: A review. *International Journal of Advances in Soft Computing and its Applications, 7*(3), 176–204.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175–185.

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R.G., van Rijn, J.N., & Vanschoren, J. (2017). OpenML benchmarking suites and the OpenML100. arXiv:1708.03731

Branco, P., Torgo, L., & Ribeiro, R. P. (2019). Pre-processing approaches for imbalanced distributions in regression. *Neurocomputing, 343,* 76–99.

Breiman, L. (1996). Stacked regressions. *Machine Learning, 24*(1), 49–64.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16,* 321–357.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794.

Dash, R., Paramguru, R. L., & Dash, R. (2011). Comparative analysis of supervised and unsupervised discretization techniques. *International Journal of Advances in Science and Technology, 2*(3), 29–37.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pp. 194–202. Elsevier.

Gonzalez, D.L., Chen, Z., Tetteh, I.K., Pansombut, T., Semazzi, F., Kumar, V., Melechko, A., & Samatova, N.F. (2012). Hierarchical classifier-regression ensemble for multi-phase non-linear dynamic system response prediction: Application to climate analysis. In *2012 IEEE 12th international conference on data mining workshops*, pp. 781–788. IEEE.

Grinberg, N. F., Orhobor, O. I., & King, R. D. (2020). An evaluation of machine-learning for predicting phenotype: Studies in yeast, rice, and wheat. *Machine Learning, 109*(2), 251–277.

Halawani, S.M., Albidewi, I.A., & Ahmad, A. (2012). A novel ensemble method for regression via classification problems. Expert Systems with Applications

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*(3), 299–314.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis, 6*(5), 429–449.

Koleti, A., Terryn, R., et al. (2017). Data portal for the library of integrated network-based cellular signatures (LINCS) program: Integrated access to diverse large-scale cellular perturbation response data. *Nucleic Acids Research, 46*(D1), D558–D566.

Mendes-Moreira, J., Soares, C., Jorge, A. M., & Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *ACM Computing Surveys, 45*(1), 1–40.

Olier, I., Sadawi, N., Bickerton, G. R., Vanschoren, J., Grosan, C., Soldatova, L., & King, R. D. (2018). Meta-QSAR: A large-scale application of meta-learning to drug design and discovery. *Machine Learning, 107*(1), 285–311.

Orhobor, O.I., Soldatova, L.N., & King, R.D. (2020). Federated ensemble regression using classification. In *International conference on discovery science*, pp. 325–339. Springer.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

Rish, I., et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 22, pp. 41–46 (2001).

Silla, C. N., & Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery, 22*(1–2), 31–72.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288.

Torgo, L., & Ribeiro, R. (2007). Utility-based regression. In *European conference on principles of data mining and knowledge discovery*, pp. 597–604. Springer.

Torgo, L., Branco, P., Ribeiro, R. P., & Pfahringer, B. (2015). Resampling strategies for regression. *Expert Systems, 32*(3), 465–476.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.