Check for
updates

Resampling strategies for imbalanced regression: a survey and empirical analysis

Juscimara G. Avelino[1] · George D. C. Cavalcanti[1] · Rafael M. O. Cruz[2]

Abstract

Imbalanced problems can arise in different real-world situations, and to address this, certain strategies in the form of resampling or balancing algorithms are proposed. This issue has largely been studied in the context of classification, and yet, the same problem features in regression tasks, where target values are continuous. This work presents an extensive experimental study comprising various balancing and predictive models, and wich uses metrics to capture important elements for the user and to evaluate the predictive model in an imbalanced regression data context. It also proposes a taxonomy for imbalanced regression approaches based on three crucial criteria: regression model, learning process, and evaluation metrics. The study offers new insights into the use of such strategies, highlighting the advantages they bring to each model's learning process, and indicating directions for further studies. The code, data and further information related to the experiments performed herein can be found on GitHub: https://github.com/JusciAvelino/imbalancedRegression.

Keywords Imbalanced tasks · Imbalanced regression · Resampling strategies

1 Introduction

Imbalanced datasets are often encountered in multiple real-world applications. For classification tasks, such an issue has been studied (Haixiang et al. 2017; Krawczyk 2016; Johnson and Khoshgoftaar 2019). Nonetheless, it is also present in regression tasks (Branco et al. 2016). Branco et al. (2017) define imbalanced problems based on the simultaneity

✉ Juscimara G. Avelino
jga2@cin.ufpe.br

George D. C. Cavalcanti
gdcc@cin.ufpe.br

Rafael M. O. Cruz
rafael.menelau-cruz@etsmtl.ca

[1] Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, Recife, PE 50740-560, Brazil

[2] École de technologie supérieure, Université du Québec, 1100 Notre Dame St. W., Montreal, QC H3C 1K3, Canada

of two factors: (i) a disproportionate preference of the user at the domain of the target variable, and (ii) insufficient representation of the data available in the most relevant cases for the user. In classification tasks, an imbalanced dataset is determined through the presence of a class having a smaller representation (minority class) than another one (majority class). However, in regression problems, the target value is continuous, thus representing a complex definition, because the target value is not constrained to a limited set of discrete values, unlike in classification problems where the target value represents specific categories or classes. Figure 1 presents the distribution and frequency of examples drawn from an imbalanced dataset (FuelCons) with target values ranging from 2.7 to 17.3. To analyze this range, we employed a bin width of approximately 0.2, resulting in a total of 74 bins. The values at the chart's edges show little frequency and are considered rare examples. In this context, Ribeiro (2011) proposes the concept of a relevance function which determines the relevance of continuous target values in defining certain examples as rare and others as normal. This definition allows to verify an imbalanced between instances considered rare and those seen as normal.

Standard regression tasks assume that all values of the domain are of equal importance, and are typically evaluated based on the performance of the most frequent values. However, values that are little represented are often extremely relevant, not only to the user, but also in the prediction process. For example, in the context of software engineering prediction mistakes in large projects are associated with higher development costs (Rathore and Kumar 2017), whereas during temperatures prediction in a meteorological application, errors that surface while predicting extreme conditions (e.g., very high temperatures) are even much more costly (Ribeiro and Moniz 2020). This scenario presents particular difficulties for learning algorithms, which tend to follow the interval of values in greater quantity while neglecting the rare ones in the distribution. Hence, failing to obtain a good prediction performance for these particular examples.

Studies looking at solutions for imbalanced regression problems have faced relatively little scrutiny when compared to those related to classification problems (Haixiang et al. 2017). The most common approach used to address this gap has been to modify the distribution of examples by balancing the training data before the actual learning process begins. Some of these strategies are Random Under-sampling (Torgo et al. 2013), which removes examples from intervals having greater quantities, Random Over-sampling (Branco et al. 2019), which replicates rare values in the dataset, and the WEighted Relevance-based Combination Strategy (WERCS) (Branco et al. 2019), which creates a weighted combination biased versions of the under- and over-sampling strategies. In addition, several real-world imbalanced regression problems rely on resampling strategies to properly deal with rare and extreme cases, such as in software defect prediction (Bal and Kumar 2018, 2020;

Fig. 1 Distribution and frequency of the target value Y from the FuelCons dataset
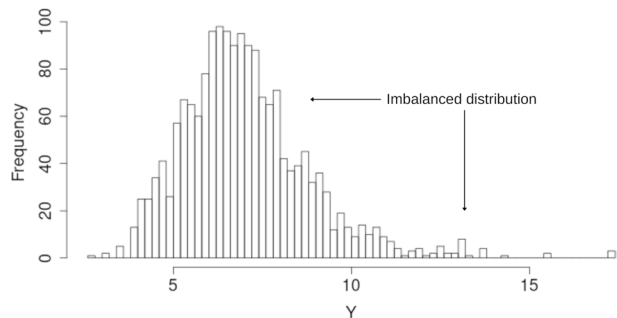
Rathore and Kumar 2017 and Rathore and Kumar 2017) and Enzyme Optimum Temperature prediction (Gado et al. 2020), as well as to assist in detecting arsenic concentration in soil using satellite imagery (Agrawal and Petersen 2021). Hence, the variety of problems and increased interest in this field demonstrates the need for studies on imbalanced regression techniques.

Another difficulty encountered in such scenarios is related to the fact that traditional performance metrics, such as the Mean Squared Error (MSE) and the Mean Absolute Error (MAE), do not adequately capture user-defined criteria (Branco et al. 2019). Additionally, recent works have proposed new performance metrics for evaluating the performance of regression models under imbalanced target distributions, and place greater emphasis on errors occurring in rare cases. In these cases, Precision, Recall, and F1-score metrics, as described for regression tasks (Torgo and Ribeiro 2009), and the squared error-relevance area (SERA) metric proposed in Ribeiro and Moniz (2020), are commonly used. Nevertheless, a comparison between multiple imbalanced regression strategies under these performance metrics, and of how they differ in their approach to assessing the model's performance, is still an open question.

Therefore, our main goal is to analyze the effects of resampling strategies for dealing with imbalanced regression problems from different perspectives. To this end, we conduct an extensive experimental study employing different resampling strategies and learning algorithms. In addition, we use metrics that can assess the models' performance in imbalanced regression tasks, such as the F1-score for regression and SERA (Ribeiro and Moniz 2020). To the best of our knowledge, this is the first work that performs a comprehensive empirical analysis of resampling techniques for imbalanced regression tasks. In contrast, for imbalanced classification tasks, numerous surveys and empirical studies have evaluated resampling algorithms in different scenarios, such as binary problems (García et al. 2020; Kovács 2019; Wojciechowski and Wilk 2017; Roy et al. 2018; Ali et al. 2019; Del Rio et al. 2015; Díez-Pastor et al. 2015; Moniz and Monteiro 2021), multiclass classification (Cruz et al. 2019; Sáez et al. 2016), and data streams (Aguiar et al. 2022; Zyblewski et al. 2019).

The broad scope of our experimental analysis, which considers multiple resampling strategies, regression models, and performance metrics, is at the core of the uniqueness of our research since it allowed us to assess the relationship among these three variables. Our study thus differs from Branco et al. (2016), which addresses only theoretical aspects of imbalanced problems in general. Moreover, regarding the performance metrics, using the SERA metric (Ribeiro and Moniz 2020) is highlighted since no other work has evaluated all resampling strategies using it specifically.

The following research questions guide this study: (i) Is it worth using resampling strategies? (ii) Which resampling strategies influence predictive performance the most? (iii) Does the choice of best strategy depend on the problem, the learning model, and the metrics used? (iv) Does the number of training examples resulting from each strategy influence the results? (v) Do the features of the data (percentage of rare cases, number of rare cases, dataset size, number of attributes and imbalance ratio) impact the predictive performance of the models? The experimental analysis revealed that resampling strategies are beneficial to the vast majority of regression models. The best strategies include GN, RO, and WERCS. Another important point is that choosing the best strategy depends on the dataset, the regression model, and the metric used when evaluating the system's performance. Furthermore, we found that the dataset size, the number of rare cases, the number of attribute and the imbalance ratio significantly influence the results. The smallest datasets and those with the fewest rare cases are the most challenging. Models demonstrate superior performance

in datasets with fewer features. Lastly, concerning the imbalance ratio, regression models encounter more significant challenges with a higher imbalance ratio.

Contributions

- We propose a novel taxonomy for imbalanced regression tasks according to the regression model, learning strategy and metrics.
- We review the main strategies used for imbalanced regression tasks.
- We conduct an extensive experimental study comparing the performance of state-of-the-art resampling strategies and their effects on multiple learning algorithms and novel performance metrics proposed in the literature.
- We analyze the impact of dataset characteristics (e.g., dataset size and the number of rare cases) on the model's predictive performance.

This work is organized as follows: Sect. 2 presents the basic concepts and proposes a taxonomy for imbalanced regression problems. Section 3 describes the resampling approaches evaluated in this study highlighting their advantages and disadvantages. Section 4 presents the experimental methodology by describing the data, algorithms, parameters, and performance metrics used in this work. Results are shown in Sect. 5. Section 6 presents the lessons learned by revisiting and answering the research questions. Finally, Sect. 7 brings our conclusions.

2 Basic concepts and proposed taxonomy

Some fundamental concepts must be grasped in order to understand the notion of imbalanced regression. In this context, the concept of relevance function is presented herein and a taxonomy is proposed to organize the strategies required. The relevance function is a fundamental concept in imbalanced regression, as it defines the importance of each sample in the dataset. Finally, a taxonomy is proposed to categorize the approaches used to address imbalanced regression problems, providing a way to understand the existing literature. Based on this taxonomy, we review the main strategies for dealing with imbalanced regression problems.

2.1 Relevance function

The concept of relevance function is crucial when it comes to understanding the imbalanced regression problem and some strategies for dealing with it. Proposed by Ribeiro (2011), the relevance function ($\phi : Y \to [0, 1]$) determines the relevance of the examples in each dataset using an automatic method. The relevance value determines the examples that are normal and those that are rare, with the rare ones being the least represented in the dataset. The intuition of the relevance function is to automatically set the significance of data points within a dataset by assigning relevance scores. In this way, the relevance function serves as the foundation for evaluating models in the context of imbalanced regression, as well as for data resampling. Consequently, using a different relevance function alters both the model evaluation and data resampling.

To the best of our knowledge, this definition of relevance function is unique in the literature. In Ribeiro (2011) and Ribeiro and Moniz (2020), the relevance function is showcased using the Piecewise Cubic Hermite Interpolating Polynomials (pchip) and

cubic spline methods. However, it was noted that cubic spline interpolation cannot provide precise control over the function. It fails to confine the relevance function within the specified [0, 1] interval scale. This limitation is rectified by the pchip method, employing suitable derivatives at control points, thereby ensuring properties like positivity, monotonicity, and convexity. Consequently, Ribeiro (2011) proposed relevance function utilizes the pchip method and aligned with this, the works in the field utilize this function.

The relevance function (ϕ) is calculated using Piecewise Cubic Hermite Interpolating Polynomials (pchip) (Dougherty et al. 1989) over a set of control points (Algorithm 1). The algorithm receives as input the control points (S) with their respective relevance values ($\varphi(y_k)$) and derivative ($\varphi'(y_k)$). The condition $y1 < y2 < \ldots < ys$ ensures that the data points are ordered in ascending order of their y-values. This ordering is fundamental for properly functioning the pchip algorithm. As a result, the algorithm produces a separate $\phi(y)$ polynomial for each interval $[y_k, y_{k+1}]$, with coefficients calculated based on the control points and their derivatives within that specific interval, where the variable k represents the index for the input set S control points.

Algorithm 1 pchip(S): Piecewise cubic Hermite interpolating polynomials

---

Input:
$S = \{\langle y_k, \varphi(y_k), \varphi'(y_k)\rangle\}_{k=1}^{s}$, with $y_1 < y_2 < \ldots < y_s$, relevance values $\varphi(y_k)$ and preliminary derivative $\varphi'(y_k)$

Output:
$\phi(y)$: a Piecewise Cubic Hermite Interpolating Polynomial
1: **for** $k \leftarrow 1$ **to** $s - 1$ **do**
2:     $h_k \leftarrow y_{k+1} - y_k$
3:     $\delta_k \leftarrow (\varphi(y_{k+1}) - \varphi(y_k))/h_k$
4:     $a_k \leftarrow \varphi(y_k)$
5: **end for**
6: $\{b_k\}_{k=1}^{s-1} \leftarrow check\_slopes\ (\{\varphi'(y_k)\}_{k=1}^{s-1}, \{\delta_k\}_{k=1}^{s-1})$ ▷ Monotone Cubic Spline
7: **for** $k \leftarrow 1$ **to** $s - 1$ **do**
8:     $c_k \leftarrow (3\delta_k - 2b_k + b_{k+1})/h_k$
9:     $d_k \leftarrow (b_k - 2\delta_k + b_{k+1})/h_k^2$
10: **end for**
11: **return** $\phi(y) = a_k + b_k(y - y_k) + c_k(y - y_k)^2 + d_k(y - y_k)^3, y \in [y_k, y_{k+1}]$

The control points can be defined based on domain knowledge or provided by an automated method. When control points are defined based on domain knowledge, selecting them is guided by the expertise and understanding of the specific problem or dataset. This approach relies on the insights and experience of individuals familiar with the data and its context. Ideally, access to domain knowledge for defining control points would be preferred. However, this knowledge is often unavailable or nonexistent (Ribeiro and Moniz 2020). Therefore, the utilization of an automatic method for control point definition becomes necessary. An example of defining control points of the NO_2 emissions problem based on domain knowledge is presented in Table 1. Control points

Table 1 Control points of *LNO*2 concentration thresholds according to Directive 2008/50/EC (Ribeiro and Moniz 2020)

y_k: *LNO*2 concentration values		$\phi(y_k)$	$\phi'(y_k)$
Low concentration	$ln(3\mu g/m^3) \approx 1.1$	0.0	0.0
Annual mean guideline	$ln(40\mu g/m^3) \approx 3.7$	0.0	0.0
Limit threshold	$ln(150\mu g/m^3) \approx 5.0$	1.0	0.0

are determined based on Directive 2008/50/EC. The objective is to maintain the *LNO*2 (target) hourly concentration values below a limit equal to $ln(150\mu g/m^3) \approx 5.0$, indicating maximum relevance, and the annual average guideline of $\ln(40\mu g / m^3) \approx 3.7$, indicating minimal relevance. And the lowest *LNO*2 concentration value $\ln(3\mu g / m^3) \approx 1.1$ is attributed minimal relevance.

In this work, we employ the automatic method, proposed by Ribeiro (2011), to define the control points. This method is based on Tukey's boxplot (Tukey 1970). The Tukey's boxplot is a graphical representation used to display the distribution of a dataset through its five summary statistics: The adjacent limits adj_L (Eq. 1) and adj_H (Eq. 2), first quartile ($Q1$), third quartile ($Q3$) and median $\tilde{Y}$ (Eq. 3). In turn, the control points are defined by the adjacent limits and the median value. The input to the pchip algorithm consists of control points, their relevance and derivatives. For this purpose, to the adjacent values (adj_L, adj_H) maximum relevance is assigned, which equals 1, and the median value ($\tilde{Y}$) with relevance value equal to zero. All control points are initialized with derivative $\phi'(y_k)$ equal to 0. In addition to defining the control points using Tukey's boxplot, Ribeiro and Moniz (2020) proposes the utilization of the adjusted boxplot, as proposed by Hubert and Vandervieren (2008).

$$adj_L = Q1 - 1.5 \cdot IQR \qquad (1)$$

$$adj_H = Q3 + 1.5 \cdot IQR \qquad (2)$$

$$\tilde{Y} = \text{ median of } Y \qquad (3)$$

where $Q1$ and $Q3$ are the first and third quartile, respectively, and $IQR = Q3 - Q1$.

Figure 2 illustrates the relevance function resulting from the pchip algorithm, for the fuelCons dataset. The points approaching $\tilde{Y}$ have negligible relevance, whereas points that move away from $\tilde{Y}$ and approach adj_L or adj_H have maximum relevance.

Fig. 2 Relevance function of the fuelCons dataset
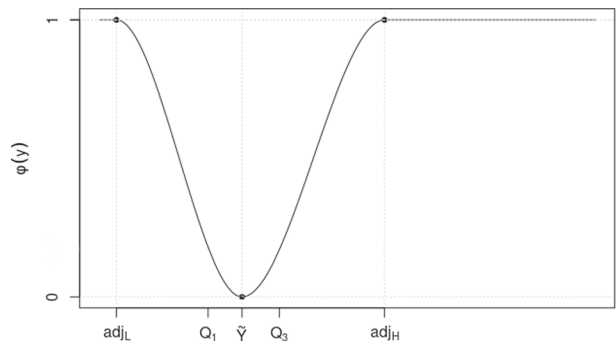
Algorithm 2 check_slopes (Φ, Δ) Fritsch and Carlson (1980)

Input:
   $\Phi = \{\varphi'(y_k)\}_{k=1}^{s-1}, \Delta = \{\delta_k\}_{k=1}^{s-1}$

Output:
   Φ: Modified derivative values

1: **for** $k \leftarrow 1$ **to** $s-1$ **do**
2:      **if** $\delta_k = 0$ **then**   ▷ Initialize the derivatives
3:          $\varphi'(y_k) \leftarrow \varphi'(y_{k+1}) \leftarrow 0$
4:      **else**
5:          $\alpha \leftarrow \varphi'(y_k)/\delta_k$
6:          $\beta \leftarrow \varphi'(y_{k+1})/\delta_k$
7:          **if** $\varphi'(y_k) \neq 0 \wedge \alpha < 0$ **then** ▷ select an underset to preserve monotonicity
8:              $\varphi'(y_k) \leftarrow -\varphi'(y_k)$
9:              $\alpha \leftarrow \varphi'(y_k)/\delta_k$
10:          **end if**
11:          **if** $\varphi'(y_{k+1}) \neq 0 \wedge \beta < 0$ **then** ▷ select an underset to preserve monotonicity
12:              $\varphi'(y_{k+1}) \leftarrow -\varphi'(y_{k+1})$
13:              $\beta \leftarrow \varphi'(y_{k+1})/\delta_k$
14:          **end if**
15:          $\tau_1 \leftarrow 2\alpha + \beta - 3$
16:          $\tau_2 \leftarrow \alpha + 2\beta - 3$
17:          **if** $\tau_1 > 0 \wedge \tau_2 > 0 \wedge \alpha(\tau_1 + \tau_2) < \tau_1\tau_2$ **then** ▷ modifying the derivative values
18:              $\tau \leftarrow 3\delta_k/\sqrt{\alpha^2 + \beta^2}$
19:              $\varphi'(y_k) \leftarrow \alpha\tau$
20:              $\varphi'(y_{k+1}) \leftarrow \beta\tau$
21:          **end if**
22:      **end if**
23: **end for**
24: **return** $\Phi = \{\varphi'(y_k)\}_{k=1}^{s-1}$

The interpolation generates a function that crosses the control points. One of the main goals is to learn the correct slopes in the data points such that the interpolant is monotonic by parts. To this end, a method that implements the Monotone Cubic Spline (Fritsch and Carlson 1980) (line 6) is used. The check_slopes method (Algorithm 2) ensures that the derivative is zero when the control point for a maximum or minimum local (Ribeiro and Moniz 2020).

A relevance threshold (t_R) defined by the user is employed to divide the data into rare (D_R) and normal (D_N) values. Given a dataset D, the sets D_R and D_N are defined considering the superior and inferior thresholds as follows: $D_R = \{\langle \mathbf{x}, y \rangle \in D : \phi(y) \geq t_R\}$ and $D_N = \{\langle \mathbf{x}, y \rangle \in D : \phi(y) < t_R\}$.

2.2 Proposed taxonomy

In the context of class imbalanced problems, solutions are often classified into four groups: Algorithmic level, Cost-sensitive, Ensemble learning, and Data preprocessing (Galar et al.

2011; López et al. 2013). However, one problem with this classification is that there is a significant overlap between the ensemble learning, data preprocessing, and cost-sensitive groups. Ensemble learning approaches can be used in conjunction with any other approaches by learning the base models, accounting to target imbalance at the algorithmic level, or applying data preprocessing prior to training each base model in the ensemble. Therefore, to better understand the different approaches for dealing with imbalanced regression problems, we can categorize the strategies into three main groups: (i) Regression Models, (ii) Learning Process Modification, and (iii) Evaluation Metrics.

The first group of strategies comprises regression models, such as single models and ensembles, which can be used to address imbalanced regression problems. However, their performance can be further improved by incorporating data preprocessing, cost-sensitive learning, and algorithmic-level modifications. The second group describes these additional strategies which can help adjust the learning process to deal with the target imbalance, thus leading to better results when compared to using the models alone. The third group comprises the evaluation metrics and is divided into local and global subgroups. The local metrics require a relevance threshold to distinguish extreme values and conduct a local evaluation, and thus, cases with a relevance score lower than the threshold are disregarded. Conversely, global metrics do not require a relevance threshold, making a global evaluation, considering all the examples. To conclude, categorizing these strategies into three groups can provide a better understanding of the approaches and enable the selection of the most suitable strategy for dealing with imbalanced regression problems. As shown in Fig. 3, data preprocessing takes the spotlight, which is the main focus of this work. Herein, we explore and compare different data preprocessing techniques to improve the performance of regression models (single models and ensembles) in imbalanced regression problems.

2.2.1 Regression models

Regression models such as MLPRegressor, Linear Support Vector Regression (SVR) and decision trees can be used to solve problems with imbalanced regression data, but they may not perform well due to the imbalance. In such cases, it may be necessary to utilize
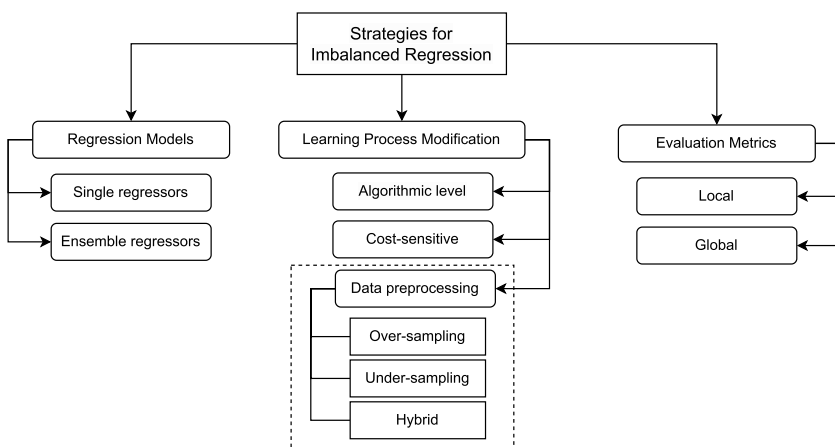

Fig. 3 Proposed taxonomy for imbalanced regression problems

other techniques such as data preprocessing or cost-sensitive learning, or to modify the algorithm, to address the issue. In the same perspective, ensemble models, such as bagging, boosting, and random forest, can also be utilized in addressing these problems. Solutions based on ensemble learning combined with data preprocessing strategies and cost-sensitive were proposed. In Branco et al. (2018) the REsampled BAGGing (REBAGG) model was proposed in a bid to integrate data resampling strategies with bagging, and had the advantage of generating a diverse set of models taking into account the different ways training data are resampled using the Random Under-sampling, Random Over-sampling and SmoteR strategies. SMOTEBoost (Moniz et al. 2018) includes a resampling step when boosting, where SmoteR is used to direct the distribution of data towards rare cases. In the same context, Moniz et al. (2017) carried out a performance study of ensemble methods in regression tasks with imbalanced datasets.

2.2.2 Learning process modification

Learning Process Modification refers to the techniques used to modify the training process of machine learning algorithms to take into account rare cases. These techniques include algorithmic level modification, as well as the cost-sensitive and data preprocessing methods. At an algorithmic level, a model is introduced in Torgo and Ribeiro (2003) with new division criteria for the regression trees that allow to induce trees at extreme and rare predicted values. Yang et al. (2021) proposed methods aimed at favoring the similarity between near targets by applying a kernel distribution to soften the distribution in the target and space of attributes. Ribeiro (2011) then addressed a utility-based algorithm involving cost-sensitive learning designed with a set of rules extracted from the generation of different regression trees aimed at obtaining accurate and interpretable predictions for imbalanced regression. Steininger et al. (2021) proposed a density-based weighting approach to address the issue of imbalanced regression, building on the cost-sensitive method. This approach assigns higher weights to rare cases by taking into account their local densities. Finally, one of the most common approaches for treating imbalanced issues is data preprocessing, also known as resampling or balancing algorithms, which precede the learning process, altering the examples distribution. The method works by either removing samples from common cases (i.e., under-sampling) or generating synthetic samples for rare events (i.e., oversampling). Data processing techniques have the advantage of allowing the use of just about any learning algorithm concurrently, without affecting the explicability of the model (Branco et al. 2019).

Different resampling strategies have been proposed to deal with imbalanced regression problems. Most such techniques are based on existing resampling strategies proposed for classification problems. That is the case, for example, of the SmoteR algorithm, which is a variation of the Smote (Chawla et al. 2002) algorithm, with the following main adaptations made to adjust to the issue of regression: (i) the definition of rare cases, (ii) the creation of synthetic examples, and (iii) the definition of target values for newly generated examples. Also on the basis of the Smote algorithm, Camacho et al. (2022) proposed Geometric SMOTE, which generates synthetic data points along the line connecting two existing data points. Other strategies adapted from imbalanced classification are: Random Under-sampling (Torgo et al. 2013), based on the idea of Kubat et al. (1997); Random Over-sampling (Branco et al. 2019), proposed for the classification in Batista et al. (2004),

and Introduction of Gaussian Noise (Branco et al. 2019), adapted from Lee (1999, 2000). In contrast, the SMOGN (SmoteR with Gaussian Noise) (Branco et al. 2017) and the WERCS (WEighted Relevance-based Combination Strategy) (Branco et al. 2019) strategies were originally proposed for handling imbalanced regression problems. Furthermore, Song et al. (2022) introduced a distributed version of the SMOGN called DistSMOGN. The method uses a weighted sampling technique to generate synthetic samples for rare cases, in addition to considering the data distribution in each node of the distributed system. For the imbalanced data streams in regression models context, Aminian et al. (2021) introduced two sampling strategies (ChebyUS, ChebyOS) based on the Chebyshev inequality to improve the performance of existing regression methods on imbalanced data streams. The approaches use a weighted learning strategy that assigns higher weights to rare cases in order to balance the training process.

Each strategy resamples data differently. However, they appear to be based on the same principles: reducing normal examples and/or increasing rare examples. Under-sampling, which reduces normal examples, is the basis of the Random Under-sampling strategy. In contrast, over-sampling, which increases rare examples, can have a simple performance, as in Random Over-sampling, or by generating synthetic cases, as in the SmoteR Algorithm and Introduction of Gaussian Noise. Other strategies are based on the aforementioned models. Examples include the SmoteR with Gaussian Noise (SMOGN), which combines the Random Under-sampling strategy with the SmoteR and Introduction of Gaussian Noise over-sampling strategies. Also, the WEighted Relevance-based Combination Strategy (WERCS) combines the Random Under-sampling and Random Over-sampling strategies by using weights to perform the resampling without establishing a relevance threshold.

In our study, we analyze a variety of data preprocessing techniques to optimize the performance of single and ensemble regression models in addressing imbalanced regression problems. Our objective is to compare the effectiveness of different approaches in identifying the most suitable strategies for this situation. By carefully assessing these techniques, we aim to provide guidance as to how to increase the success rate of regression models using data preprocessing techniques in imbalanced regression tasks.

2.2.3 Evaluation metrics

The choice of assessment metrics is fundamental in an imbalanced datasets scenario. Some metrics, such as the MSE, may fool users when the focus is on the accuracy of rare values of the target variable (Moniz et al. 2014) since it does not consider the relevance of each testing example. To show the limitations of the MSE metric and how the scores obtained by different metrics can significantly differ, we present a synthetic example (Table 2). For 10 examples in the *FuelCons* dataset, we present hypothetical predictions for two artificial models: M_1 and M_2. The True row represents the true target for each instance in the dataset, directly obtained from the FuelCons dataset. The ϕ row is the relevance value of each example. Meanwhile, the M_1 and M_2 rows showcase predictions generated by the respective models for individual test examples. In parallel, the M_1 and M_2 loss rows quantify the differences between the true target and the predictions made by the models for each test example. The example shows that M_1 generates more accurate predictions for the less relevant examples, which are less represented in the dataset, while M_2 performs better for more relevant examples, which are more frequently represented. Nonetheless, if the models' performances are assessed using the MSE metric, there will be no difference in scores between them. This is because the MSE metric

Table 2 Predictions of two artificial models

	Test examples									
True	2.70	3.20	3.50	4.10	4.50	4.70	5.20	5.70	9.20	17.30
ϕ	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.02**	**0.57**	**1.00**	**1.00**	**1.00**
M_1	2.66	3.14	3.40	3.80	4.00	3.80	4.10	4.40	7.70	15.50
M_1 loss	**0.04**	**0.06**	**0.10**	**0.30**	**0.50**	**0.90**	**1.10**	**1.30**	**1.50**	**1.80**
M_2	0.90	1.70	2.20	3.00	3.60	4.20	4.90	5.60	9.14	17.26
M_2 Loss	**1.80**	**1.50**	**1.30**	**1.10**	**0.90**	**0.50**	**0.30**	**0.10**	**0.06**	**0.04**

True—Target values

ϕ—Relevance values

M1 and M2—Model predictions

M1 Loss and M2 Loss—Prediction errors

considers all examples as having the same relevance (ϕ). Therefore, for the imbalanced data scenario, where each example has a particular relevance, it is more interesting to use metrics that consider the relevance of each particular example.

Other metrics consider each example as having a particular relevance score, such as Precision, Recall, and the F1-score, which were proposed for regression applications in Torgo and Ribeiro (2009). In addition, the Squared error-relevance area (SERA) metric, which was specifically created for imbalanced regression, was proposed by Ribeiro and Moniz (2020). This metric aims to effectively assess the model's performance for predictions of extreme values while being robust to model bias. Table 3 presents the MSE, F1-score, and SERA values for the example presented in Table 2. As earlier mentioned, for the MSE, the models are regarded as equals since they both have the same error amplitude. Nonetheless, for the F1-score and SERA, which consider each example's relevance, M_2 is the best model as it presents a lower error in the most important examples.

The Precision, Recall, and F1-score metrics require that a relevance threshold be defined to determine extreme values. Thus, a local evaluation is performed, since examples below the threshold are ignored. Furthermore, these metrics use the concept of a utility-based framework (Torgo and Ribeiro 2007; Ribeiro 2011). Such a structure uses the numeric error of the prediction and the relevance of the actual and predicted values. The utility of predicting a value $\hat{y}$ for y is calculated from the notions of costs and benefits of numeric predictions (Branco et al. 2019), and thus, the utility function $U_\phi^p(\hat{y}, y)$ is given by Eq. 4, where $\hat{y}$ is the predicted value and y is the actual value.

$$
\begin{aligned}
U_\phi^p(\hat{y}, y) &= B_\phi(\hat{y}, y) - C_\phi^p(\hat{y}, y) \\
&= \phi(y) \cdot (1 - \Gamma_B(\hat{y}, y)) - \phi^p(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y)
\end{aligned}
\tag{4}
$$

Table 3 Performances of two artificial models

	Estimated performance		
	MSE	F1-score	SERA
M_1	0.955	0.598	7.885
M_2	0.955	0.983	0.076

The utility is given by the difference between the prediction benefit $(B_\phi(\hat{y}, y))$ and cost $(C_\phi^p(\hat{y}, y))$ of prediction $\hat{y}$ for y. The benefit is defined as a proportion of the relevance of the actual value according to the following equation: $\phi(y) \cdot (1 - \Gamma_B(\hat{y}, y))$, where $\Gamma_B(\hat{y}, y)$ is the bounded loss function (Eq. 5). This equation defines a loss function, $\Gamma_B(\hat{y}, y)$, which quantifies the loss incurred when making a prediction $\hat{y}$ for the actual value y (Eq. 6). This loss function operates on a scale from 0 to 1, where 0 represents no loss, and 1 represents maximum loss.

$$\Gamma_B(\hat{y}, y)) = \begin{cases} L(\hat{y}, y)/\dot{L}_B(\hat{y}, y), & \text{if } L(\hat{y}, y) < \dot{L}_B(\hat{y}, y) \\ 1, & \text{if } L(\hat{y}, y) \geq \dot{L}_B(\hat{y}, y) \end{cases} \tag{5}$$

L is a "standard" loss function [e.g., absolute deviation (Eq. 6)] and $\dot{L}_B$ is the benefit threshold function, (Eq. 7). The benefit threshold function identifies the point at which the predicted value ceases to provide a benefit. This can happen because of two conditions: (i) surpassing the maximum acceptable loss of the bump or (ii) being situated on a different bump (Ribeiro 2011).

$$L(\hat{y}, y) = |\hat{y} - y| \tag{6}$$

$$\dot{L}_B(\hat{y}, y) = min\{b_{\gamma(y)}^\Delta, \ddot{L}_B(\hat{y}, y)\} \tag{7}$$

where $b_{\gamma(y)}^\Delta$ is the maximum admissible loss, defined in Eq. 8. The maximum admissible loss is calculated for each bump i. A bump refers to a interval of the domain, denoted as $B \subseteq Y$ (Ribeiro 2011). b^- is the mean value at which the target variable reaches the minimum relevance before reaching its maximum value, and b^* is the mean value at which the target variable reaches the maximum relevance. The reason for this definition is that this function is contingent upon the smallest discrepancy concerning the target variable when transitioning from the most pertinent value within a bump (b_i^*) to an alternative bump. The smallest differences regarding the target variable can have two effects on model performance. On the positive side, it can make the model more accurate by focusing on the areas where predictions must be very close to the actual values. This is useful when you need high accuracy in specific parts of the data. Conversely, the model might become too fixated on the training data, making it sensitive to unusual data points and not very good at handling new data, leading to overfitting. Consequently, this implies that when dealing with "narrow" bumps, our sensitivity to prediction errors is heightened, whereas for broader bumps, we are more inclined to deem larger disparities between the actual and forecasted values as acceptable (Ribeiro 2011).

$$b_{\gamma(y)}^\Delta = 2 \cdot \min \{| b_i^- - b_i^* |, | b_i^* - b_{i+1}^- |\} \tag{8}$$

Figure 4 shows the bump partition obtained for a relevance function and the maximum admissible loss for each bump. This arbitrary relevance function, defined in the context of non-uniform utility regression, has four quite different bumps.

And $\ddot{L}_B(\hat{y}, y))$ (Eq. 9) is defined as follows:

$$\ddot{L}_B(\hat{y}, y)) = \begin{cases} | y - b_{\gamma(y)}^- |, & \text{if } \hat{y} < y \\ | y - b_{\gamma(y)+1}^- |, & \text{if } \hat{y} \geq y \end{cases} \tag{9}$$
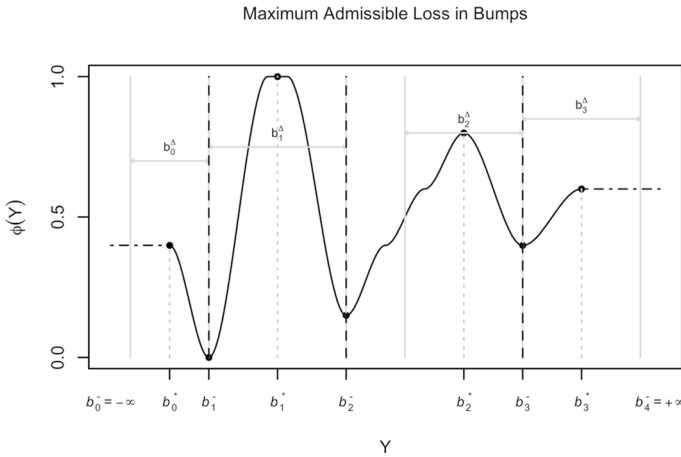
Maximum Admissible Loss in Bumps

Fig. 4 Bumps partition of Y with respect to relevance function ϕ and the maximum admissible loss in bumps. Each bump i is characterized by its partition node b^- and by one global maximum b^*. Each bump has a maximum error tolerance defined by the double of the smalles amplitude in the bump between each of one of its bounds and its maximum value (Ribeiro 2011)

This definition satisfies two essential conditions: (1) The initial component within the min function addresses the maximum allowable error range within the true value's context, guaranteeing a level of reasonable accuracy in the prediction; (2) The subsequent component within the min function evaluates whether the predicted value aligns with the correct action by considering its proximity to the boundaries of the context associated with the true value.

The cost is given by the mean of weighted relevance ($\phi^p(\hat{y}, y)$) (Eq. 10), where the parameter p is used to define the weights between the two relevances and $\Gamma_C(\hat{y}, y)$ is the bounded loss function in the scale [0;1]. This equation calculates the weighted relevance of the predicted value $\hat{y}$ and the actual value y. The parameter p defines the weights between these two relevances. The intuition here is to balance the predicted value's importance and the utility function's actual value.

$$\phi^p(\hat{y}, y) = (1 - p)\phi(\hat{y}) + p\phi(y) \qquad (10)$$

The cost function $\Gamma_C(\hat{y}, y)$ is calculated according to Eq. 11.

$$\Gamma_C(\hat{y}, y)) = \begin{cases} L(\hat{y}, y)/\dot{L}_C(\hat{y}, y), & \text{if } L(\hat{y}, y) < \dot{L}_C(\hat{y}, y) \\ 1, & \text{if } L(\hat{y}, y) \geq \dot{L}_C(\hat{y}, y) \end{cases} \qquad (11)$$

where L is the standard loss function, and $\dot{L}_C$ is the cost threshold function (Eq. 12):

$$\dot{L}_C(\hat{y}, y) = min\{b^\Delta_{\gamma(y)}, \ddot{L}_C(\hat{y}, y)\} \qquad (12)$$

and $\ddot{L}_C(\hat{y}, y))$ is defined as follows:

$$\ddot{L}_C(\hat{y}, y)) = \begin{cases} \mid y - b^*_{\gamma(y)-1} \mid, & \text{if } \hat{y} < y \\ \mid y - b^*_{\gamma(y)+1} \mid, & \text{if } \hat{y} \geq y \end{cases} \qquad (13)$$

Captured using the utility function, the Precision and Recall metrics are defined by Eqs. 14 and 15, respectively.

$$Precision = \frac{\sum_{\phi(\hat{y}_i)>t_R}(1 + U_\phi^p(\hat{y}_i, y_i))}{\sum_{\phi(\hat{y}_i)>t_R}(1 + \phi(\hat{y}_i))} \tag{14}$$

$$Recall = \frac{\sum_{\phi(y_i)>t_R}(1 + U_\phi^p(\hat{y}_i, y_i))}{\sum_{\phi(y_i)>t_R}(1 + \phi(y_i))} \tag{15}$$

The relevance of the actual value y_i is defined by $\phi(y_i)$, as defined in Sect. 2.1, and $\phi(\hat{y}_i)$ is the relevance of the predicted value $\hat{y}_i$. t_R is a threshold defined by the user for the relevance values, and $U_\phi^p(\hat{y}_i, y_i)$ is the utility function previously described.

The Precision and Recall metrics can be aggregated in compound measures, such as F1-score, defined by Eq. 16:t

$$F1\text{-}score = \frac{(\beta^2 + 1) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \tag{16}$$

where $0 \leq \beta \leq 1$ controls the relative importance of the Recall for the Precision. These compound measures have the advantage of allowing comparisons between models by providing a single score (Torgo and Ribeiro 2009).

These metrics require the definition of an ad-hoc relevance threshold and do not consider examples below the threshold for model evaluation (Ribeiro and Moniz 2020). To address this, Ribeiro and Moniz (2020) proposed the SERA metric.

SERA metric can assess models' efficacy and optimize them for predicting rare and extreme cases. This metric does not require a definition of a relevance threshold and thus performs a global evaluation since all data points are considered. The Squared error-relevance is obtained in relation to a cutting t achieved based on a relevance function $\phi : Y \to [0, 1]$. A subset $D^t = \{\langle \mathbf{x}, y \rangle \in D : \phi(y) \geq t\}$ formed based on the cutting t is considered for this estimate, such as in Eq. 17:

$$SER_t = \sum_{i \in D^t}(\hat{y}_i - y_i)^2 \tag{17}$$

The Squared error-relevance area (SERA) represents the area below the curve SER_t, obtained through integration presented in Eq. 18:

$$SERA = \int_0^1 SER_t \, dt = \int_0^1 \sum_{i \in D^t}(\hat{y}_i - y_i)^2 \, dt \tag{18}$$

The SER_t curve offers a broad view of prediction errors in the domain at various relevance cutoff values. Therefore, a smaller area under the curve (SERA) indicates a better model. It is noteworthy that assuming uniform preferences with $\phi(y) = 1$, SERA is comparable with the sum of squared errors.

3 Resampling strategies

The most common way to deal with imbalanced datasets is to use resampling strategies changing the data distribution to balance the targets (Moniz et al. 2017). Such strategies are concentrated on the following three main approaches: (i) over-sampling, (ii) under-sampling, and (iii) a combination of these two approaches. In over-sampling, rare cases are generated to compensate for the imbalanced distribution. The Random Over-sampling technique (Branco et al. 2019) is an example of such a technique, which works by replicating rare cases prior to training. However, it is also possible to perform over-sampling by generating synthetic cases, as in the SmoteR (Torgo et al. 2013) and Introduction of Gaussian Noise strategies (Branco et al. 2019).

Conversely, under-sampling techniques aim to exclude larger quantity data (i.e., normal examples). The Random Under-sampling algorithm (Torgo et al. 2013) uses this notion. Some strategies employ a combination of approaches, such as the SmoteR and Introduction of Gaussian Noise, which generates synthetic cases and uses under-sampling, WEighted Relevance based Combination Strategy (Branco et al. 2019), thus combining the approaches of under-sampling and over-sampling. The SMOGN (Branco et al. 2017) uses the generation of synthetic cases with SmoteR and GN and under-sampling.

Sections 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6 provide an overview of the resampling strategies evaluated in this work. These strategies were selected based on their wide adoption in the literature. Conversely, other strategies were disregarded due to an absence of publicly available source code for them, limited reproducibility, and infrequent utilization by researchers for diverse problem domains. Finally, Sect. 3.7 critically analyzes the resampling strategies with a visual example.

3.1 SmoteR

The SMOTE for regression (SmoteR) algorithm was proposed in Torgo et al. (2013) (Algorithm 3). Like the other methods addressing imbalanced regression issues, it requires a relevance function ($\phi(y)$) and a relevance threshold (t_R). The relevant or unimportant examples are defined from such a function. The algorithm removes the least relevant examples (lines 4 to 7), which are considered "normal", and then generates synthetic examples based on the most relevant examples (line 8). The generation process basically follows the idea in the SMOTE, namely, first selecting one rare case from the dataset as the seed case and one of its K-Nearest Neighbors to generate a new data point between the reference and its selected neighbor. Algorithm 4 presents the procedure for generating the synthetic cases using SmoteR. First the number of synthetic examples that is generated from a selected rare case, *ng*, is determined based on the percentage of over-sampling *o* determined by the user and the dataset cardinality |D| (line 3). Then, for each rare case *c* that will be used as a reference in the generation process, its K-Nearest Neighbors are computed (Line 5) *nns*. After the set of neighbors are obtained, the algorithms execute multiple iterations to generate *ng* synthetic examples by picking one of the examples in the *nns* set at random and interpolating with the reference one. This generation process is presented from lines 8 to

15, which show how attribute values for the synthetic case are generated. If the attributes are numeric, the difference between the attributes of the two seed cases is calculated (line 10). Subsequently, (line 11) multiplies this difference by a random number between 0 and 1, and then adds to the example's attribute. Otherwise, a random selection between the values of the seed cases is performed. On lines 16 to 18, the value of the target is generated, calculated by the weighted average of the two cases. The weights are obtained by the distance between the new case and the two seed cases (lines 16 and 17). In de Oliveira Branco (2018), this strategy is extended, and is able to handle any number of either normal or rare cases.

Algorithm 3   SmoteR

Input:
D - original dataset
t_R - relevance threshold
o - percentage of over-sampling
u - percentage of under-sampling
k - the number of neighbors used in case generation
Output:
$newD$ - a new modified dataset

1: $Bins_R \leftarrow \{D_i \in D : \forall (x,y) \in D_i, \phi(y) \geq t_R\}$ ▷ partitions into relevant bins
2: $Bins_N \leftarrow D \backslash Bins_R$ ▷ partitions into normal bins
3: $newD \leftarrow Bins_R$
4: **for each** $B \in Bins_N$ **do**
5:     $selNormCases \leftarrow$ random sample of $u \times |B|$ cases of B ▷ under-sampling procedure
6:        $newD \leftarrow newD \bigcup selNormCases$
7: **end for**
8: $newCases \leftarrow GENSYNTHCASES(Bins_R, o, k)$ ▷ Generation of the attribute values
9: $newD \leftarrow newD \bigcup newCases$
10: **return** $newD$

Algorithm 4 Generating synthetic cases

Input:
  D - original dataset
  o - percentage of over-sampling
  k - the number of neighbors used in case generation

1: **function** GENSYNTHCASES(D, o, k)
2:   $newCases \leftarrow \{\}$
3:   $ng \leftarrow (o - 1) \times |D|$ ▷ number of new cases to generate for each existing case
4:   **for all** $case \in D$ **do**
5:     $nns \leftarrow kNN(k, case, D_R \backslash \{case\})$ ▷ k-Nearest Neighbours of case
6:     **for** $i \leftarrow 1$ to ng **do**
7:       $x \leftarrow$ randomly choose one of the nns
8:       **for each** $a \in$ attributes **do**   ▷ generation of the attribute values
9:         **if** ISNUMERIC(a) **then**
10:           $diff \leftarrow case[a] - x[a]$
11:           $new[a] \leftarrow case[a] + RANDOM(0, 1) \times diff$
12:         **else**
13:           $new[a] \leftarrow$ randomly select among case[a] and $x[a]$
14:         **end if**
15:       **end for**   ▷ generation of the target value
16:       $d_1 \leftarrow DIST(new, case)$
17:       $d_2 \leftarrow DIST(new, x)$
18:       $new[y] \leftarrow \frac{d_2 \times case[y] + d_1 \times x[y]}{d_1 + d_2}$
19:     **end for**
20:     $newCases \leftarrow newCases \bigcup new$ ▷ add the new synthetic case
21:   **end for**
22:   **return** $newCases$
23: **end function**

3.2 Random over-sampling

The Random over-sampling (Branco et al. 2019) strategy, presented in Algorithm 5, works by first selecting the examples that are above the relevance threshold t_R (line 2) as candidates to be duplicated, $Bins_R$. Then, for each bin B belonging to the rare examples $Bins_R$, the number of replicas *tgtNr* generated is defined according to its cardinality $|B|$ and the oversampling percentage o (Line 4). The $|B|$ represents the number of elements (data points or examples) contained within that specific bin B. This oversampling percentage is a hyperparameter defined by the user. Random sampling is performed on line 5, and the duplicated cases are added to the new dataset (*newD*) on line 6. When performing this algorithm, no special treatment is required to generate the target values. As the examples generated are identical to the existing rare cases, the duplicated ones have exactly the same target value.

Algorithm 5   Random over-sampling

Input:
     D - original dataset
     t_R - relevance threshold
     o - percentage of over-sampling
Output:
     $newD$ - a new modified dataset
1:   $newD \leftarrow D$
2:   $Bins_R \leftarrow \{D_i \in D : \forall (x, y) \in D_i, \phi(y) \geq t_R\}$
3:   **for each** $B \in Bins_R$ **do**
4:      $tgtNr \leftarrow o \times |B|$ $\triangleright$ number of replicas to be added
5:      $selCases \leftarrow$ sample randomly $tgtNr$ elements from B
6:      $newD \leftarrow newD \bigcup selCases$ $\triangleright$ add the replicas to the new data
7:   **end for**
8:   **return** $newD$

Algorithm 6   Random under-sampling

Input:
     D - original dataset
     t_R - relevance threshold
     u - percentage of under-sampling
Output:
     $newD$ - a new modified dataset
1:   $Bins_R \leftarrow \{D_i \in D : \forall (x, y) \in D_i, \phi(y) \geq t_R\}$
2:   $Bins_N \leftarrow D \backslash Bins_R$
3:   $newD \leftarrow Bins_R$
4:   **for each** $B \in Bins_N$ **do**
5:      $tgtNr \leftarrow u \times |B|$ $\triangleright$ number of replicas to be removed
6:      $NormCases \leftarrow$ randomly under-sample $tgtNr$ elements from B $\triangleright$ remove the examples from B
7:      $newD \leftarrow newD \bigcup NormCases$
8:   **end for**
9:   **return** $newD$

3.3 Random under-sampling

The Random Under-Sampling strategy (Algorithm 6) was proposed by Torgo et al. (2013). In this approach, the under-sampling is performed by first using the relevance function (Sect. 2.1) and a relevance threshold t_R to define the rare cases in the dataset (line 1). The

examples below t_R are considered normal, being candidates to be removed from the final dataset (Branco et al. 2016) (line 2), while rare cases are kept. The removal of the normal examples is thus performed according to an under-sampling rate provided by the user u, which defines the percentage of under-sampling applied in the dataset. For each bin B belonging to the set of normal examples $Bins_N$, the number of examples removed from it is computed based on its cardinality and the percentage of undersampling u (Line 5). Line 6 performs the under-sampling in B by randomly selecting data points to be removed, resulting in a reduced set that is used to compose the final dataset *newD*.

3.4 Introduction of Gaussian noise

Generating synthetic examples through Gaussian noise (Introduction of Gaussian Noise - GN) constitutes an adaptation of the method proposed in Lee (1999, 2000) for classification tasks to the regression context. Algorithm 7 presents the GN technique. It starts by dividing the dataset into normal cases $Bins_N$ and rare cases $Bins_R$ according to the relevance function $\phi(y)$ and the relevance threshold t_R (Lines 1 and 2). Examples belonging to $Bins_N$ (i.e., normal examples) are reduced in size, using the Random under-sampling technique (lines 4 to 6). The amount of reduction is controlled by the percentage of the under-sampling hyperparameter u defined by the user.

From lines 8 to 20, the over-sampling procedure is performed using the samples in $Bins_R$. For each seed case selected and used in the generation process, a total of ng new artificial generated examples are added to the dataset. ng is computed based on the percentage of the overs-sampling hyperparameter o and the number of examples in the corresponding set $B \in Bins_R$ (Line 9). The artificial cases are generated by introducing a small perturbation on both the attributes and the target variable value of the seed case. If the attributes are nominal (line 13), the generation is performed with probability proportional to the frequency of the values found in the category (lines 14 and 15). Otherwise, for the numeric attributes, a random perturbation from a normal distribution is added, as indicated on lines 17 and 18, where δ is the perturbation amplitude defined by the user and $sd(a)$ is the standard deviation of the attribute a estimated using the examples in the category. The normal perturbation is also applied to the seed target value in order to generate the target value of the newly generated example.

Algorithm 7 Introduction of Gaussian Noise

Input:
    D - original dataset
    t_R - relevance threshold
    u - percentage of under-sampling
    o - percentage of over-sampling
    δ - perturbation amplitude
Output:
    $newD$ - a new modified dataset

1: $Bins_N \leftarrow \{D_i \in D : \forall(x,y) \in D_i, \phi(y) < t_R\}$
2: $Bins_R \leftarrow \{D_i \in D : \forall(x,y) \in D_i, \phi(y) \geq t_R\}$
3: $newD \leftarrow Bins_R$
4: **for each** $B \in Bins_N$ **do**
5:     $selNormCases \leftarrow$ random sample of $u \times |B|$ elements from B
6:     $newD \leftarrow newD \bigcup selNormCases$
7: **end for**
8: **for each** $B \in Bins_R$ **do**   ▷ over-sampling procedure
9:     $ng \leftarrow o \times |B|$ ▷ number of synthetic examples for each case in B
10:     **for each** $case \in B$ **do**   ▷ generate synthetic examples
11:         **for** $i \leftarrow 1$ **to** ng **do**
12:             **for each** $a \in Attrs \bigcup Y$ **do**
13:                 **if** a is $nominal$ **then**
14:                     $probs \leftarrow$ frequency of possible values of a
15:                     $new[a] \leftarrow$ sample a value from the values of a with weights $=$
    $probs$
16:                 **else**
17:                     $new[a] \leftarrow$ multiply $case[a]$ with a random sample from
18:                     $N(0, \delta \cdot sd(a))$
19:                 **end if**
20:             **end for**
21:             $newD \leftarrow newD \bigcup \{new\}$ ▷ add synthetic case to newD
22:         **end for**
23:     **end for**
24: **end for**
25: **return** $newD$

3.5 SmoteR with Gaussian noise

The SmoteR with Gaussian Noise (SMOGN - SG) (Branco et al. 2017) (Algorithm 8) combines the Random under-sampling strategy (lines 6 to 9) with two over-sampling strategies: SmoteR and Introduction of Gaussian Noise. The goal is to limit the potential risks to the SmoteR of generating bad examples when the seed and its selected neighbor

are not close enough by using the more conservative strategy of just introducing Gaussian noise to generate new cases. These bad examples may not represent of the underlying data distribution and can introduce several issues like noise, bias, or inconsistencies into the dataset. Moreover, the technique aims to allow for an increase in diversity when generating examples, which is not feasible by using only the Introduction of Gaussian Noise method (Branco et al. 2017). Increasing diversity means producing a comprehensive range of examples covering different data distribution aspects. The generated examples should not be overly similar or redundant. Instead, they should capture different patterns, variations, or scenarios present in the data to represent the data distribution comprehensively. Thus, SMOGN addresses the main drawbacks of SmoteR and the introduction of Gaussian noise techniques.

Line 11 determines the number of synthetic cases *ng* that will be generated according to the percentage of the over-sampling hyperparameter *o* and the number of existing cases in the corresponding bin *B*. Then, for each seed case in *B*, its K-Nearest Neighbors and the maximum allowed distance to generate new cases with SmoteR are computed (lines13 to 15). When the seed case and the selected neighbor are "sufficiently near" (i.e., distance below the computed threshold *maxD*), the SMOGN generates new synthetic examples with the SmoteR (lines 17 and 18) technique. Otherwise, it uses the Introduction of Gaussian Noise method when the distance between the two examples is higher than the estimated threshold (lines 20 and 21). The generated data points are then added to the new dataset, *newD*.

Algorithm 8 SMOGN

Input:
  D - original dataset
  t_R - relevance threshold
  u - percentage of under-sampling
  o - percentage of over-sampling
  k - number of nearest neighbors
  $dist$ - distance metric
Output:
  $newD$ - a new modified dataset

1: $OrdD \leftarrow D$ order D by ascending value of Y
2: $\phi() \leftarrow$ relevance function
3: $Bins_N \leftarrow$ partitions of consecutive examples $\langle x_i, y_i \rangle \in OrdD$, such that $\phi(y_i) < t_R$
4: $Bins_R \leftarrow$ partitions of consecutive examples $\langle x_i, y_i \rangle \in OrdD$, such that $\phi(y_i) \geq t_R$
5: $newD \leftarrow Bins_R$
6: **for each** $B \in Bins_N$ **do**   ▷ under-sampling procedure
7:     $selNormCases \leftarrow$ randomly sample $u \times |B|$ cases from B
8:     $newD \leftarrow newD \bigcup selNormCases$
9: **end for**
10: **for each** $B \in Bins_R$ **do**   ▷ over-sampling procedure
11:     $ng \leftarrow o \times |B|$ ▷ number of synthetic examples for each case in B
12:     **for each** $case \in B$ **do**   ▷ generate synthetic examples
13:         $nns \leftarrow kNN(k, case, dist)$ ▷ K-Nearest Neighbors of case
14:         $DistM \leftarrow$ distances between the case and the examples in B
15:         $maxD \leftarrow median(DistM)/2$
16:         **for** $i \leftarrow 1$ **to** ng **do** $x \leftarrow$ randomly choose one of the nns
17:             **if** $DistM(x) < maxD$ **then**   ▷ safe kNN selected
18:                 $new \leftarrow$ use SmoteR to interpolate x and $case$
19:             **else**   ▷ non-safe kNN selected
20:                 $pert \leftarrow min(maxD, 0.02)$
21:                 $new \leftarrow$ introduce Gaussian Noise in $case$ with a perturbation $pert$
22:             **end if**
23:             $newD \leftarrow newD \bigcup new$ ▷ add synthetic case to newD
24:         **end for**
25:     **end for**
26: **end for**
27: **return** $newD$

3.6 WEighted relevance based combination strategy

The WEighted Relevance-based Combination Strategy (WERCS) strategy (Branco et al. 2019) combines biased versions of the under- and over-sampling strategies which depend exclusively on the relevance function provided to the dataset without requiring establishing a relevance threshold. Under the WERCS, the relevance function and a modification of the relevance are used to attribute weights that are used as inclusion and removal criteria for the examples. Algorithm 9 details this resampling strategy. The over-sampling and

under-sampling on lines 4 and 7, respectively, are performed considering weights obtained on lines 3 and 6. These weights are calculated based on the relevance function. The weights associated with over-sampling *WOver* are proportional to the relevance function (line 3). Therefore, the higher the relevance of a case, the higher its probability of being selected for generating new cases. Conversely, the weights associated with under-sampling *WUnd* are inversely proportional to the relevance value (line 6). Thus, normal examples, which are usually associated with lower relevance values, have a higher probability of being removed rather than used in the generation process. The number of generated and removed samples is defined based on the percentage of over-sampling *o* and under-sampling *u*, respectively.

Therefore, the main advantage of this technique is that as a relevance threshold is not set a priori, each example can participate in both processes. Thus, both under-sampling and over-sampling strategies are applied over the entire dataset. Also, the technique eliminates the dependency on the relevance threshold t_R that was a key component necessary for applying all other resampling strategies reviewed in this work.

Algorithm 9 WEighted relevance-based combination strategy (WERCS)

Input:
  D - original dataset
  u - percentage of under-sampling
  o - percentage of over-sampling
Output:
  $newD$ - a new modified dataset
1: $\phi() \leftarrow$ relevance function
2: $newD \leftarrow D$
3: $WOver \leftarrow \{\phi(y_i) \mid y_i \in Y\}$
4: $Over \leftarrow$ sample $o \times |D|$ cases from D with $WOver$ weights   ▷ over-sampling procedure
5: $newD \leftarrow newD \bigcup Over$
6: $WUnd \leftarrow \{1 - \phi(y_i) \mid y_i \in Y\}$
7: $Und \leftarrow$ sample $u \times |D|$ cases from D with $WUnd$ weights
8: $newD \leftarrow newD \backslash Und$ ▷ under-sampling procedure
9: **return** $newD$

3.7 Advantages and disadvantages of strategies

The strategies to resample data can have both advantages and disadvantages. Therefore, it is crucial to understand the behavior of each strategy. While these strategies can potentially enhance learning, they can also impede the learning process of the models. Figure 5 introduces the result of applying the resampling strategies to the *FuelCons* dataset. The following values were attributed to the algorithm's parameter: *u/o* = balance and $t_R = 0.8$ (except for the WERCS, since it does not require establishing the threshold). The standard values were adopted for the remaining parameters. For the visualization, the target values (Y) and the attribute (X30) were considered.

Despite selecting the nearest examples to generate new cases, SmoteR still involves the risk of the example being too far and of generating an example that does not
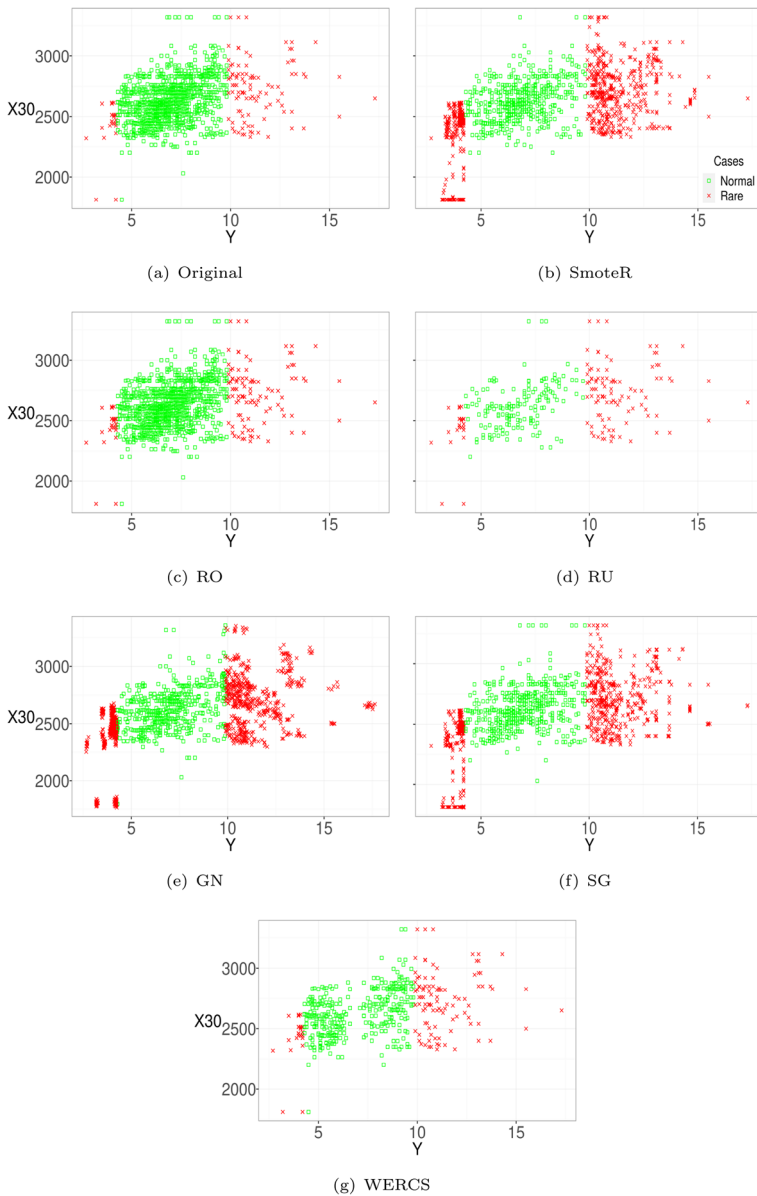
Fig. 5 Distribution of the examples of the FuelCons dataset after applying the resampling strategies, considering $t_R=0.8$

correspond to the seed very well. This phenomenon is shown in the lower left side of Fig. 5b, where the generated examples are far from the original examples. In the RO strategy, high percentages of over-sampling may cause an overfitting (Branco et al. 2019) problem. Even though the technique increases the representation of rare cases considerably, the generated dataset does not present a high points diversity. The

generation process consists in just duplicating existing samples without covering the feature space well.

Figure 5c shows the rare data points in darker shade, given that the RO only makes copies of the examples. This can therefore lead to learning algorithms overfitting such rare examples. In addition, if the replication rate is too high, many duplicate data points are added to the dataset which can significantly increase the training time. In contrast to the RO, in the RU strategy some meaningful information may be lost due to the removal of training data (Fig. 5d), which may hamper the learning of the model. Figure 5e shows the result after using the GN strategy, which promotes over-sampling by adding normally distributed noise. Once again, in contrast to the RO strategy, examples different from the originals ones are generated, and this diversity can help to mitigate overfitting. For the SG strategy, even though one of its goals is to reduce the risks seen in SmoteR by creating different examples from the original, Fig. 5f shows that there is still a similarity with the SmoteR distribution. However, when compared to GN, it is evident that the diversity of generated examples is higher in SG. In the WERCS strategy (Fig. 5g), it can be seen that the green data points are divided into two groups after the under-sampling, and this result can complicate the learning process. The WERCS over-sampling strategy performs similarly to RO, where the generated data are copies of the originals; such as, no new information is added to the training set.

The advantages and disadvantages of each resampling strategy are quite evident, as is the fact that there is no perfect strategy. We hypothesize that other variables, such as the regression model and the dataset under investigation, are required to determine the best data resampling strategy. Thus, our research allows to understand the behaviors of these strategies with different regression models and problems, which in turn allows to establish directions for combinations of the three variables, namely, the resampling strategy, the regression model, and the dataset.

4 Research methodology

4.1 Datasets

Experiments were performed using 30 imbalanced regression datasets chosen to match the frequency generally used in studies looking at imbalanced regression. The levels of imbalance in these datasets are defined from the relevance function (Sect. 2.1). A study conducted by Branco et al. (2019) involved varying the relevance threshold from 0.5 to 1. Nevertheless, the findings showed a complex relationship between the number of rare cases, the learning algorithm, and the applied pre-processing strategy. Therefore, our experiments considered a commonly used threshold (t_R) of 0.8, as used in Branco et al. (2017), Branco et al. (2019) and Branco et al. (2018). Thus, we obtained datasets with different percentages of rare cases (imbalanced levels), varying between 5.1% and 23.4%. The main features of these sets are presented in Table 4. Datasets are presented in descending order in terms of the percentage of rare cases (%Rare). It is important to clarify that counting rare cases is conducted across the entire dataset, as commonly practiced in the literature. Counting rare cases on the entire dataset is crucial for comprehensively understanding their rarity within the data context. This approach allows us to analyze the model's behavior within the original context of the dataset. However, resampling strategies are applied only to the training set to prevent data leakage during cross-validation. The nominal attributes were codified,

Table 4 Characteristics of the 30 datasets used in the experiments

Datasets	N	p.total	p.nom	p.num	nRare	IR	%Rare
wine-quality	6497	11	0	11	1523	0.306	23.4
analcat-apnea3	450	11	0	11	103	0.297	22.9
meta	528	65	0	65	108	0.257	20.5
cocomo-numeric	60	56	0	56	10	0.200	16.7
Abalone	4177	8	1	7	679	0.194	16.3
a3	198	11	3	8	32	0.193	16.2
forestFires	517	12	0	12	79	0.180	15.3
a1	198	11	3	8	28	0.165	14.1
a7	198	11	3	8	27	0.158	13.6
boston	506	13	0	13	65	0.147	12.8
pdgfr	79	320	0	320	10	0.145	12.7
sensory	576	11	0	11	69	0.136	12.0
a2	198	11	3	8	22	0.125	11.1
kdd-coil-1	316	18	0	18	34	0.121	10.8
triazines	186	60	0	60	20	0.120	10.8
airfoild	1503	5	0	5	161	0.120	10.7
treasury	1049	15	0	15	109	0.116	10.4
mortgage	1049	15	0	15	106	0.112	10.1
debutanizer	2394	7	0	7	240	0.111	10.0
fuelCons	1764	37	12	25	164	0.103	9.3
heat	7400	11	3	8	664	0.099	9.0
california	20640	8	0	8	1821	0.097	8.8
AvailPwr	1802	15	7	8	157	0.095	8.7
compactiv	8192	21	0	21	713	0.095	8.7
cpuSm	8192	12	0	12	713	0.095	8.7
maxTorq	1802	32	13	19	129	0.077	7.2
lungcancer-shedden	442	24	0	24	25	0.060	5.7
space-ga	3107	6	0	6	173	0.059	5.6
ConcrStr	1030	8	0	8	55	0.056	5.3
Accel	1732	14	3	11	89	0.054	5.1

N: number of cases; p.total: number of attributes; p.nom: number of nominal attributes; p.num: number of numeric attributes; nRare: number of rare cases; Imbalance ratio (IR): $\frac{|D_R|}{|D_N|}$; %Rare: $100 \times nRare/N$. Datasets are arranged in descending order regarding the percentage of rare cases (%Rare)

transforming the vector of categories into whole values between 0 and the number of categories−1. As for the ordinal attributes, a pre-defined order was established (e.g., *small*: 1, *medium*: 2, *large*: 3).

For each dataset, the results were calculated by applying two 10-fold cross-validation repetitions (i.e., 2×10 cross-validation) in order to obtain the mean and standard deviation of the results. Nested cross-validation with 2-fold was employed to optimize the hyperparameters of the resampling strategies, specifically utilizing the SERA metric for optimization. The SERA metric was chosen to optimize the hyperparameters because it was specifically created for imbalanced regression. This metric evaluates models'

performance in predicting extreme values, penalizing model biases without requiring a threshold, and conducting a global assessment (Ribeiro and Moniz 2020). Unlike the F1-score, which conducts a local assessment by considering only rare examples, SERA evaluates all examples.

4.2 Algorithms

The experiments were performed with the following learning algorithms: Bagging (BG), Decision Tree (DT), Multilayer Perceptron (MLP), Random Forest (RF), Support Vector Machine (SVM), and XGBoost (XG). Default hyperparameters were applied for these models. For details and descriptions of default hyperparameters and used packages, refer to Online Appendix A.

As resample techniques, we considered the following strategies: SmoteR (SMT), Random Over-sampling (RO), Random Under-sampling (RU), Introduction of Gaussian Noise (GN), SMOGN (SG), and WEighted Relevance-based Combination Strategy (WERCS). Details about hyperparameters and packages can be found in Table 5.

4.3 Model evaluation

In imbalanced tasks, choosing the appropriate metrics for model evaluation is essential. This work uses the F1-score and SERA metrics to evaluate regression models, allowing the evaluation of different perspectives of the model performance. While the F1-score metric is based on the concept of utility-based evaluation and performs a local assessment according to the definition of a relevance threshold, the SERA metric evaluates the effectiveness of models in predicting extreme values while penalizing several model biases without the need for a threshold, and performing a global assessment (Ribeiro and Moniz 2020). The results for the RMSE and MAE metrics can be consulted in the supplementary material (Online Appendix B) for benchmarking purposes.

Table 5 Resampling strategies, hyperparameters, and packages used

Algorithms	Hyperparameters	Packages
SMT	u/o = {balance, extreme}, k = {3, 5, 7}	ImbalancedLearningRegression[a]
RO	o = {balance, extreme}	ImbalancedLearningRegression[a]
RU	u = {balance, extreme}	ImbalancedLearningRegression[a]
GN	u/o = {balance, extreme}, $\delta = \{0.00, 0.05, 0.10, \cdots, 0.95, 1.00\}$	ImbalancedLearningRegression[a]
SG	u/o = {balance, extreme}, k = {3, 5, 7}, $\delta = \{0.00, 0.05, 0.10, \cdots, 0.95, 1.00\}$	smogn[b]
WERCS	u, o = {0.3, 0.5, 0.7, 0.9}	resreg[c]

[a] https://pypi.org/project/ImbalancedLearningRegression/—Version 0.0.1

[b] https://pypi.org/project/smogn/—Version 0.1.2

[c] https://pypi.org/project/resreg/—Version 0.2

Table 6 Number of times each algorithm and resampling strategy achieved the best result according to the F1-score metric

	None	SMT	RO	RU	GN	SG	WERCS
BG	3	3	13	2	6	3	0
DT	6	5	8	3	6	2	0
MLP	3	4	12	1	3	6	1
RF	2	4	9	6	3	3	3
SVR	1	5	13	1	2	6	2
XG	4.7	1	7	5	5.2	3.2	4
Total	19.7	22	**62**	18	**25.2**	23.2	10

Table 7 Number of times each algorithm and resampling strategy reached the best result according to the SERA metric

	None	SMT	RO	RU	GN	SG	WERCS
BG	4	3	3	5	4	1	10
DT	4	2	7	6	5	2	4
MLP	2	4	9	1	4	7	3
RF	2	1	4	7	6	2	8
SVR	0	5	1	3	12	5	4
XG	3	1	2	5	8.5	3.5	7
Total	15	16	26	27	**39.5**	20.5	**36**

5 Results

The experiments aimed at answering the following research questions:

1. Is it worth using resampling strategies?
2. Which resampling strategies influence the predictive performance the most?
3. Does the choice of best strategy depend on the problem, the learning model, and the metrics used?
4. Does the number of training examples resulting from each strategy influence the results?
5. Do the features of the data (percentage of rare cases, number of rare cases, dataset size, number of attribues and imbalance ratio) impact the predictive performance of the models?

Tables 6 and 7 show how many times each algorithm obtains the highest value for the F1-score and SERA metrics, respectively. Where there is a tie, each of the n tied strategies receives $1/n$ point. Each row in this table must add up to 30, the number of datasets assessed. For both metrics used, we found that the larger number of wins occurs when using some of the resampling strategies, which points to an advantage of using such strategies. As highlighted in bold in the tables, RO and GN obtained the highest number of wins considering the F1-score, and the GN and WERCS, according to SERA. Another point observed is that the choice of best strategy possibly depends on the regression model used. As for the metrics, both agree regarding the GN strategy. By observing the score by rows, also in Tables 6 and 7, it is clear that there is no general agreement between the

datasets for a resampling strategy since each point is a dataset, and all of them are distributed in different strategies. The results per learning algorithm, including mean and standard deviation, can be accessed in the supplementary material – Online Appendix B.

To identify the best way to preprocess each dataset, Tables 8 and 9 introduce the best and worst results for the F1-score and SERA metrics, respectively. The results show that most datasets have distinct preferences in terms of combining the best learning model and the resampling strategy. This distinction is also found for the metrics used. As for the worst results, the SVR and MLP, without preprocessing, is the worst combinations for both metrics. Thus, balancing the dataset before applying these models is crucial to reaching more promising results. It is also crucial to note the significant difference between the best and worst results per problem. So, obtaining good results depends on the correct choice of resampling strategy and learning model. Unfortunately, the SG strategy failed to perform on the california, heat, and wine-quality datasets. These are large datasets, highlighting the potential challenges in optimizing hyperparameters, rendering the use of this model impractical.

We applied the Friedman test to better measure the advantage of using resampling strategies ($p - value < 0.05$). The Friedman statistical test was chosen since it can compare multiple techniques over several datasets (Demšar 2006). For this measurement, ranking sequences are compared. Tables 10 and 11 present the mean ranking of the means of the algorithms with a combination of each resampling strategy, considering the F1-score and SERA metrics. The lower the ranking, the better the algorithm performance. The algorithms used present significant differences. In general, the best average rank of each algorithm was obtained by using some of the resampling strategies evaluated in this work.

To verify which approaches are statistically different, we applied the Nemenyi post-hoc test. Figure 6a–f illustrates the critical difference diagrams (Demšar 2006) for each of the learning models, considering the F1-score metric. The horizontal line demonstrates the significance of the difference between the models. Models that are not connected present a significant difference ($p - value < 0.05$) in relation to the others. This test once again confirms that, globally, resampling strategies can significantly improve the regressors' performance. The Nemenyi test reveals that the RO obtained the best results and the most significant differences in relation to None (data without any preprocessing) for the metric F1-score. In most cases, the SMT, SG, RU techniques achieve the worst results. Figure 7a–f considers the SERA metric; in such a scenario, most of the best results are obtained using the GN strategy, followed by WERCS, given the number of times where the best results were achieved in the critical difference chart.

Another interesting fact is how different learning algorithms perform when no resampling strategy is applied. In both metrics, the DT model achieved better results with the original data sets. Additionally, an interesting aspect involves the ensemble models, Random Forest (RF) and XGBoost (XG) obtained better results than single models, corroborating the analysis conducted in Moniz et al. (2017), which says that ensemble methods provide a better result than single models. Conversely, the SVR and MLP algorithms obtained the worst results, especially when no preprocessing techniques were employed. Thus, it can be concluded that these algorithms are the most affected by having an imbalanced target distribution and require special attention when applied in the imbalanced regression context.

As described in Sect. 3, each resampling algorithm uses different heuristics to balance the dataset. Figure 8 illustrates the percentage of increase/decrease in the training examples for each strategy. It was previously concluded that the best results were achieved using the RO, GN, and WERCS strategies. The GN and WERCS strategies present a small

Table 8 Best and worst results for each dataset based on the F1-score metric

Datasets	Best result		Worst result	
wine-quality	0.738	RF.RO	0.000	SG[a]
analcat-apnea3	0.233	MLP.NONE	2.00e−5	SVR.NONE
meta	0.435	BG.GN	2.00e−5	SVR.NONE
cocomo-numeric	0.371	RF.RU	1.60e−5	SVR.NONE
Abalone	0.702	RF.RU	1.79e−1	SVR.NONE
a3	0.506	RF.RU	1.95e−5	SVR.NONE
forestFires	0.403	SVR.SMT	2.00e−5	SVR.NONE
a1	0.723	RF.SG	2.00e−5	SVR.NONE
a7	0.411	SVR.RO	2.00e−5	SVR.NONE
boston	0.893	RF.RO	2.00e−5	SVR.NONE
pdgfr	0.229	RF.GN	1.60e−5	RF.SG
sensory	0.672	XG.GN	2.00e−5	SVR.NONE
a2	0.580	RF.RU	2.00e−5	SVR.NONE
kdd-coil-1	0.677	RF.RU	1.90e−5	SVR.NONE
triazines	0.226	RF.WERCS	0.028	RF.NONE
airfoild	0.951	BG.RO	2.00e−5	SVR.WERCS
treasury	0.980	RF.GN	0.777	SVR.NONE
mortgage	0.985	RF.RO	0.834	SVR.NONE
debutanizer	0.901	RF.SMT	0.646	MLP.GN
fuelCons	0.942	XG.GN	0.094	MLP.RU
heat	0.989	XG.RO	0.000	SG[a]
california	0.902	XG.NONE	0.000	SG[a]
AvailPwr	0.977	XG.NONE	0.725	SVR.NONE
compactiv	0.528	RF.SG	0.109	MLP.RO
cpuSm	0.526	RF.SMT	0.105	MLP.RO
maxTorq	0.988	XG.WERCS	2.00e−5	SVR.NONE
lungcancer-shedden	0.665	MLP.SG	2.00e−5	SVR.NONE
space-ga	0.802	XG.GN	2.00e−5	SVR.NONE
ConcrStr	0.966	XG.GN	2.00e−5	SVR.RU
Accel	0.961	XG.RO	2.00e−5	SVR.GN

[a]Not completed in a reasonable timeframe

percentage of 1.28% and 2.83%, respectively. Conversely, the RO presents an increased percentage of 1421.1%. Therefore, the influence of the number of examples on the results is unclear since the strategies with different percentages of increase/decrease obtained good results. Nonetheless, it may be disadvantageous (from a training time point of view) to use a strategy, such as the RO, that considerably increases the training set. Other strategies also deliver satisfactory results without excessively increasing the training set. More details about the number of instances after the application of the resampling strategies can be found in the supplementary material – Online Appendix C.

Figures 9, 10, 11, 12 and 13 present the F1-score results arranged according to some dataset characteristics in a bid to assess their impact on the performance of the models. The following characteristics were assessed: percentage of rare cases, number of rare cases, dataset size, number of attributes, and imbalance ratio. The imbalance ratio is calculated as the ratio between the number of rare cases (D_R) and the number of normal cases (D_N), i.e.,

Table 9 Best and worst results for each dataset based on the SERA metric

Datasets	Best result		Worst result	
wine-quality	1.43e+2	RF.WERCS	0.000	SG[a]
analcat-apnea3	3.93e+7	RF.GN	4.80e+8	SVR.NONE
meta	2.79e+7	MLP.GN	3.83e+7	DT.SG
cocomo-numeric	4.16e+5	XG.SMT	2.91e+6	SVR.NONE
Abalone	1.17e+3	RF.WERCS	2.59e+3	SVR.NONE
a3	3.30e+2	SVR.GN	2.48e+7	MLP.RU
forestFires	1.88e+5	MLP.SG	3.02e+5	DT.GN
a1	1.63e+3	SVR.SG	1.76e+6	MLP.NONE
a7	2.58e+2	XG.RU	2.62e+7	MLP.RU
boston	2.30e+2	BG.WERCS	2.51e+8	MLP.RU
pdgfr	3.86e−2	RF.GN	2.40e−1	DT.WERCS
sensory	1.39e+1	RF.RU	1.16e+2	MLP.SMT
a2	6.91e+2	SVR.GN	2.28e+7	MLP.RU
kdd-coil-1	2.43e+3	SVR.RU	9.09e+3	SVR.NONE
triazines	6.99e−2	RF.RU	3.54e−1	MLP.NONE
airfoild	6.78e+7	XG.RO	2.05e+11	SVR.NONE
treasury	3.08e+0	RF.GN	1.78e+3	MLP.RU
mortgage	1.30e+0	XG.RO	1.70e+2	DT.SG
debutanizer	3.23e−1	RF.SG	2.52e+0	MLP.NONE
fuelCons	1.52e+1	XG.GN	1.32e+7	MLP.RU
heat	6.42e+2	XG.GN	0.000	SG[a]
california	1.87e+12	RF.RU	0.000	SG[a]
AvailPwr	4.72e+3	XG.NONE	1.77e+5	MLP.RU
compactiv	1.61e+3	RF.WERCS	6.26e+7	MLP.NONE
cpuSm	2.11e+3	RF.WERCS	5.25e+7	MLP.RO
maxTorq	4.63e+3	XG.GN	1.17e+6	SVR.NONE
lungcancer-shedden	5.75e+1	SVR.GN	2.16e+2	SVR.NONE
space-ga	1.78e+0	XG.WERCS	4.16e+10	MLP.RO
ConcrStr	5.64e+2	XG.WERCS	9.27e+3	SVR.GN
Accel	2.34e+1	XG.GN	7.41e+5	MLP.RU

[a]Not completed in a reasonable timeframe

$\frac{|D_R|}{|D_N|}$. Each box represents a regression model (BG, DT, MLP, RF, SVR and XG), and each point represents a specific set of data, and each line represents a resampling strategy (None, SMT, RO, RU, GN, SG and WERCS).

The results presented in Fig. 9 correspond to the same ordering provided in Table 4, where the datasets are arranged in decreasing order of the percentage of rare cases. In such conditions, it is not possible to find any pattern. Therefore, it is unclear how this aspect relates to the model's performance. Figures 10 and 11 are arranged according to the number of rare cases and the dataset size, respectively. These circumstances reveal that the smaller datasets with a lower number of rare cases represent the hardest tasks, as observed in Branco et al. (2019). Figure 12 illustrates the evolution of F1 considering the number of attributes in each dataset. In some instances, it is noticeable that datasets with fewer

Table 10 Average ranking
(F1-score)

Algorithm	Average
RF.RO	10.0
RF.GN	11.8
RF.RU	11.9
XG.RU	12.0
BG.RO	12.2
RF.WERCS	12.4
XG.WERCS	12.8
XG.GN	13.3
BG.RU	13.6
XG.RO	14.6
BG.WERCS	15.5
XG.NONE	15.6
BG.GN	16.0
RF.NONE	17.3
RF.SMT	17.8
XG.SMT	18.4
XG.SG	18.5
BG.NONE	18.8
BG.SMT	18.8
DT.RO	19.3
RF.SG	20.3
DT.NONE	20.4
BG.SG	20.5
DT.RU	20.9
DT.GN	21.0
DT.WERCS	21.1
DT.SMT	22.9
DT.SG	25.1
SVR.RO	26.5
SVR.SMT	26.7
MLP.RO	26.8
SVR.GN	27.6
SVR.SG	28.4
MLP.SMT	29.8
MLP.GN	30.8
SVR.WERCS	31.3
MLP.WERCS	31.7
MLP.SG	31.7
MLP.NONE	32.5
MLP.RU	33.3
SVR.RU	33.8
SVR.NONE	39.6

Table 11 Average ranking (SERA)

Algorithm	Average
RF.RU	8.7
RF.GN	10.0
RF.WERCS	10.9
RF.RO	11.3
XG.GN	11.5
XG.WERCS	12.4
XG.RU	12.5
BG.WERCS	13.4
RF.NONE	13.6
XG.SG	13.6
XG.NONE	13.9
BG.RU	14.2
RF.SG	14.3
BG.GN	14.7
XG.RO	14.8
BG.RO	15.4
RF.SMT	15.8
XG.SMT	17.8
BG.NONE	18.3
BG.SG	18.7
BG.SMT	19.6
SVR.SG	23.9
SVR.GN	23.9
DT.GN	26.8
DT.NONE	26.9
SVR.RO	27.1
DT.RU	27.2
DT.RO	27.3
DT.SG	27.4
SVR.RU	27.5
MLP.SG	27.8
SVR.SMT	28.0
DT.WERCS	28.6
MLP.RO	28.7
SVR.WERCS	28.9
MLP.GN	29.4
MLP.SMT	30.4
MLP.WERCS	30.7
DT.SMT	31.4
MLP.NONE	33.9
MLP.RU	35.1
SVR.NONE	36.6

(a) Results of the F1-score metric for the BG algorithm

(b) Results of the F1-score metric for the DT algorithm

(c) Results of the F1-score metric for the MLP algorithm

(d) Results of the F1-score metric for the RF algorithm

(e) Results of the F1-score metric for the SVR algorithm

(f) Results of the F1-score metric for the XG algorithm
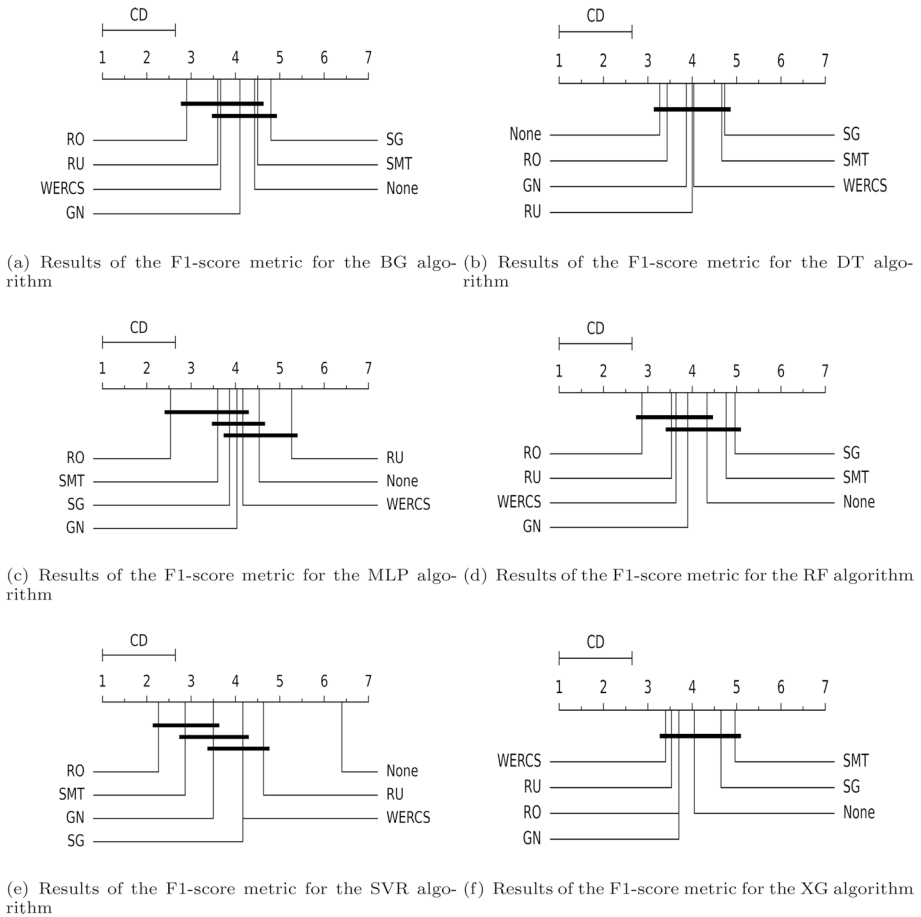
**Fig. 6** Critical difference diagrams for each learning algorithm considering the F1-score metric

features exhibit superior performance. Finally, in Fig. 13, the datasets are sorted according to their respective imbalance ratios. The regression models with all resampling strategies face more significant challenges when dealing with datasets exhibiting higher imbalance ratios. This difficulty arises because higher imbalance ratios mean the rare cases are significantly underrepresented compared to the normal cases. As a result, the model may struggle to learn the underlying patterns and become biased toward the normal cases.

For all the evaluated dataset characteristics, the behavior of the resampling strategies is quite similar, resulting in an overlap of the graph's line. For better clarity, another analysis is conducted considering the best F1-score for each dataset. The figures in Online Appendix D present the best F1-score for each dataset, considering the dataset characteristics. With this, we can visualize how the data characteristics affect the performance of the top models. The percentage of rare cases does not exhibit a clear pattern. Thus, concluding whether this characteristic affects the model's performance is challenging. Regarding the number of rare cases and the dataset size, models achieve better performance when there are more rare cases and a larger dataset. When we consider the number of attributes, we observe that a higher number leads to better model
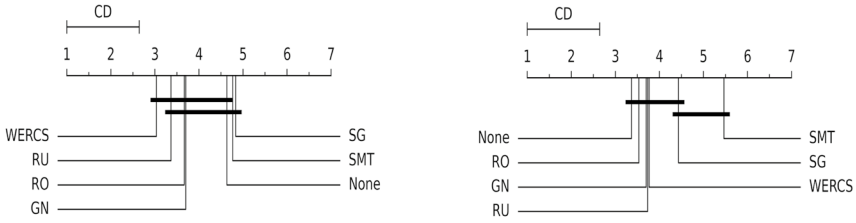
(a) Results of the SERA metric for the BG algorithm

(b) Results of the SERA metric for the DT algorithm
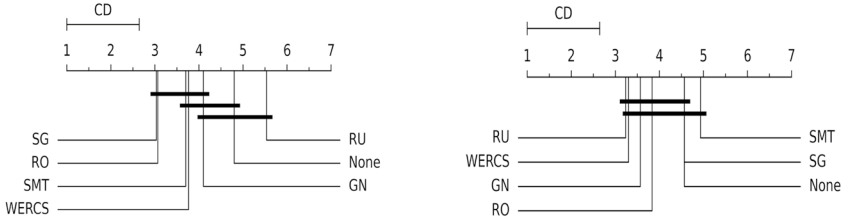
(c) Results of the SERA metric for the MLP algorithm

(d) Results of the SERA metric for the RF algorithm
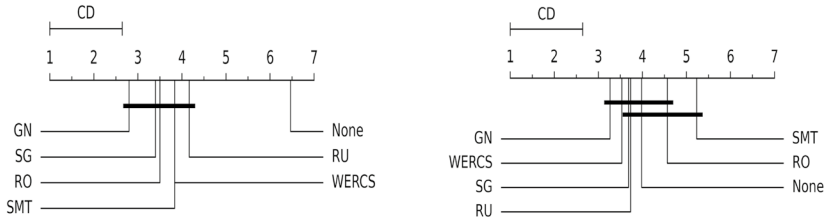
(e) Results of the SERA metric for the SVR algorithm

(f) Results of the SERA metric for the XG algorithm

Fig. 7 Critical difference diagrams for each learning algorithm considering the SERA metric
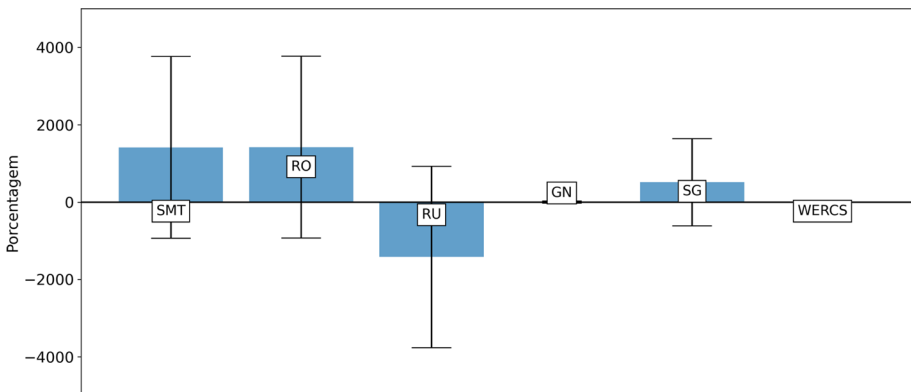
Fig. 8 Percentage of increase/decrease in the training set for each resampling strategy
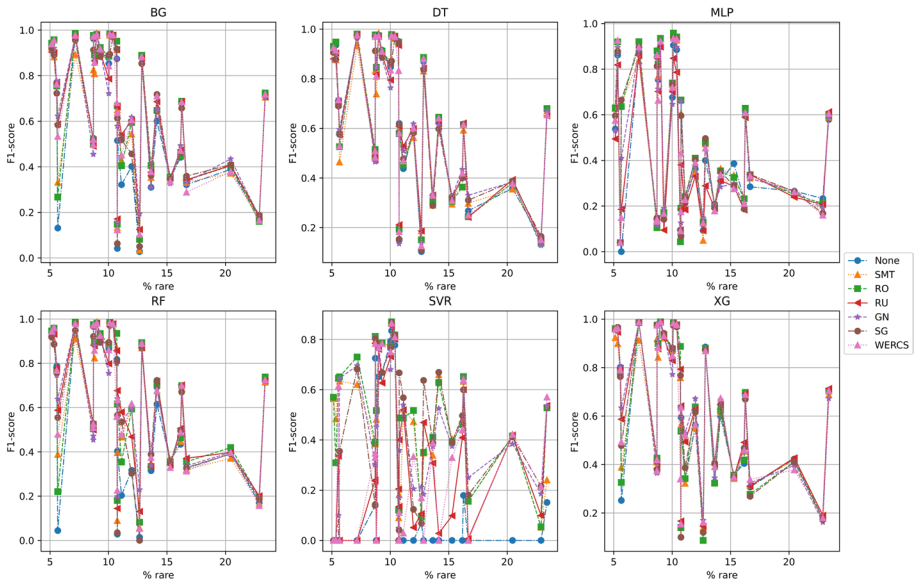
Fig. 9 Evolution of the *F1-score* with datasets sorted by percentage of rare cases
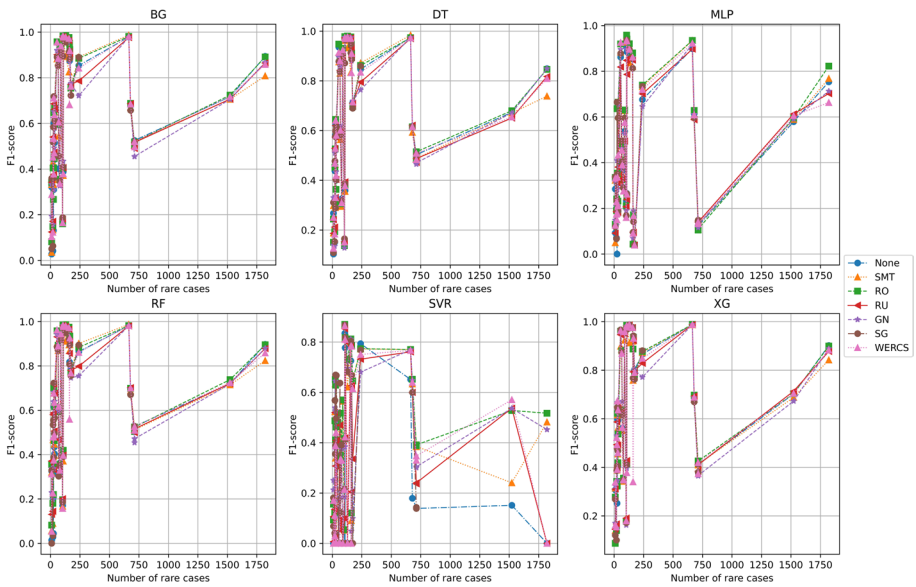
Fig. 10 Evolution of the *F1-score* with datasets sorted by number of rare cases

performance. As for the imbalance ratio, the higher the imbalance ratio, the worse the model's performance.
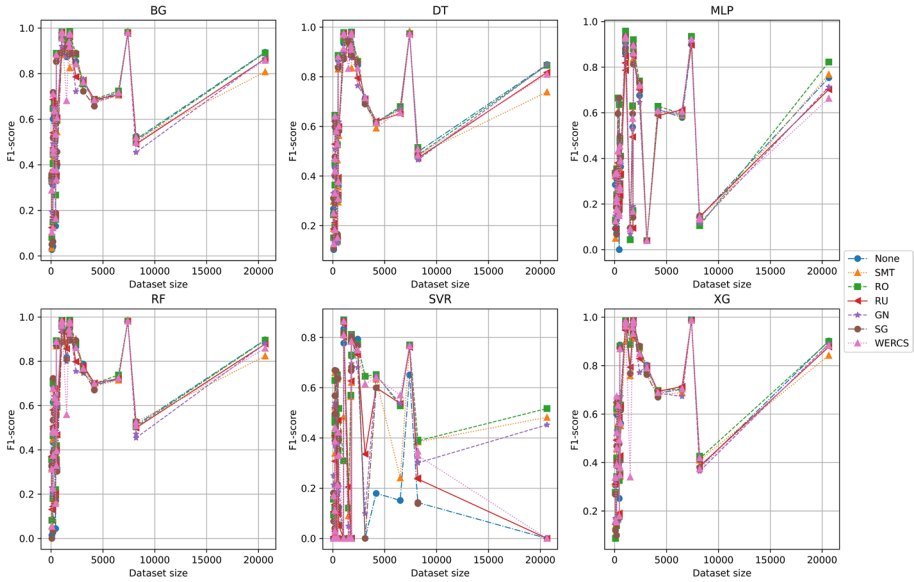
Fig. 11 Evolution of the *F1-score* with datasets sorted by size
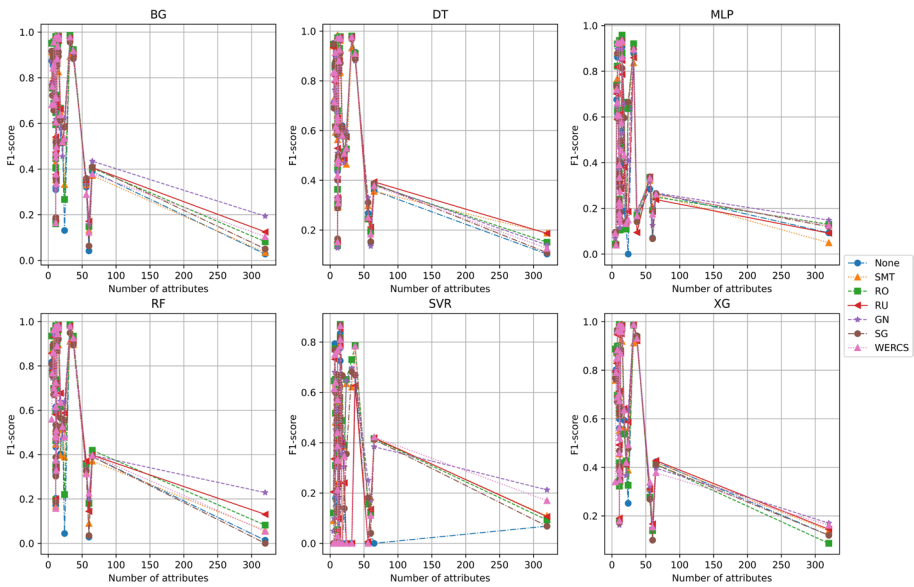
Fig. 12 Evolution of the *F1-score* with datasets sorted by number of attributes

6 Lessons learned

Different approaches have been proposed in a bid to solve the imbalanced problem in the context of regression, including resampling strategies. Our research introduced a review and an experimental study of the main resampling strategies for dealing with imbalanced
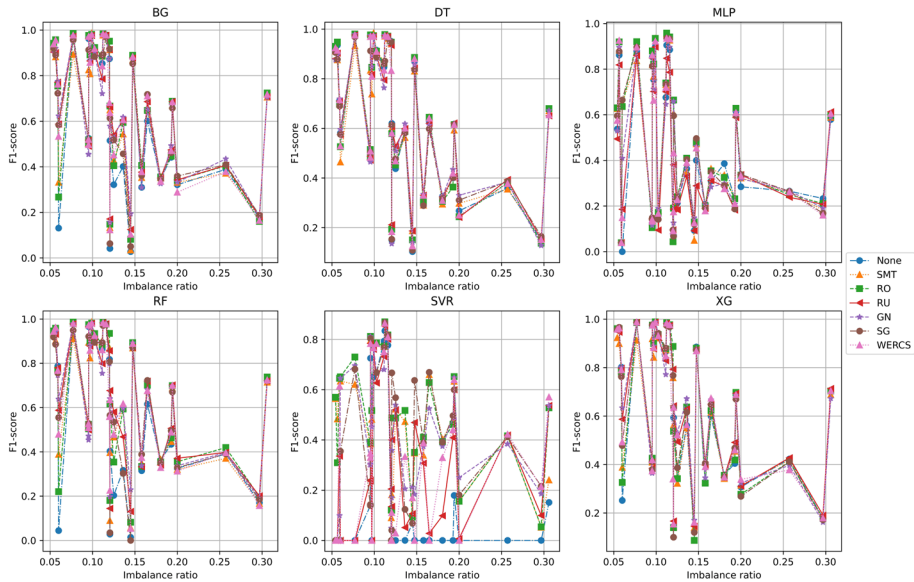
Fig. 13 Evolution of the *F1-score* with datasets sorted by imbalance ratio

regression problems. In this section, the research questions are revisited and answered succinctly.

1. *Is it worth using resampling strategies?*

     We answer this question by accounting for the number of times that each strategy won (Tables 6 and 7). For both metrics, four of the resampling strategies used won more times than when no resampling strategy was used. Furthermore, the Nemenyi post-hoc statistical tests performed (Figs. 6 and 7) demonstrate that many resampling strategies are statistically better as compared to the absence of a strategy. Therefore, it is advantageous to use (some) resampling strategies.

2. *Which resampling strategies influence the predictive performance the most?*

     Considering the F1-score metric, the RO and GN strategies positively influenced the results of the learning algorithms. As for the SERA metric, the GN and WERCS techniques are the best strategies. Statistically, in general, the GN, RO, and WERCS strategies held the best results (Figs. 6 and 7). Conversely, in terms of predictive performance, the SMT, SG, RU techniques achieve the worst results..

3. *Does the choice of best strategy depend on the dataset, the learning model, and the metrics used?*

     Most of the datasets used have distinct preferences regarding the combination of the best regression model and resampling strategy (Tables 8 and 9). For the regression models, different resampling strategies can reach better results. As for the metrics, both agree that the GN is a good resampling strategy. Nonetheless, there are cases of disagreement between them.

4. *Does the number of training examples resulting from each strategy influence the results?*

     Given that the best results were obtained using the GN, RO, and WERCS strategies, which have different percentages (1.28%, 1421.1%, 2.83%, respectively) of increase/ decrease in the training examples (Fig. 8), the influence of the number of examples on

the results is not clear. Nonetheless, it may not be advantageous (from a training time point of view) to use a strategy like the RO, which considerably increases the training set, as other strategies also deliver equivalent results without this excessive increase.

5. *Do the features of the data (percentage of rare cases, number of rare cases, dataset size, number of attribues and imbalance ratio) impact the predictive performance of the models?*

     In the studies performed, the percentage of rare cases did not have a clear impact on the results. On the other hand, considering the dataset size and number of rare cases, it could be seen that the smaller datasets with fewer rare cases correspond to the most difficult tasks. Models demonstrate superior performance in datasets with fewer features. Lastly, concerning the imbalance ratio, regression models encounter more significant challenges with a higher imbalance ratio. The results for this question are shown in Figs. 9, 10, 11, 12 and 13. Online Appendix D presents the evolution of the best F1-score for each dataset characteristic, providing a clearer view of the impact of these dataset characteristics on model performance.

7 Conclusion

This work reviews and performs a comparative study of data resampling strategies for handling imbalanced regression problems. We reviewed six state-of-the-art resampling strategies for regression based on three approaches: (i) under-sampling, (ii) oversampling, and (iii) a mix of undersampling and oversampling, while discussing the advantages and drawbacks of each existing technique.

Then, we performed an extensive experimental analysis comprised of 6 regression algorithms and 7 scenarios (6 resampling strategies and not using resampling) that can guide the development of new strategies to solve the imbalanced regression problem. Our experimental results demonstrate that it is important to use a resampling technique for most models as resampling techniques lead to statistically better results. The experimental study also shows that no resampling technique outperforms all others. Furthermore, choosing the best resampling technique depends on three main factors: the learning algorithm, the dataset, and the performance metric used to assess the model's performance.

Further studies should address the recommendation of combining resampling strategies with a regression model for each specific dataset. Another element worth addressing is the dataset characteristics, which should be investigated through data complexity measures (Lorena et al. 2018) in order to assess the adverse effects of these features on prediction performance. Moreover, an essential point to address involves proposing a new relevance function since currently, only one definition exists. This proposal aims to conduct studies and comparisons regarding the definition of an imbalanced regression dataset.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Agrawal A, Petersen MR (2021) Detecting arsenic contamination using satellite imagery and machine learning. Toxics 9(12):333

Aguiar, G., Krawczyk, B., Cano, A.: A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. arXiv preprint arXiv:2204.03719 (2022)

Ali H, Salleh MNM, Hussain K, Ahmad A, Ullah A, Muhammad A, Naseem R, Khan M (2019) A review on data preprocessing methods for class imbalance problem. Int J Eng Technol 8:390–397

Aminian E, Ribeiro RP, Gama J (2021) Chebyshev approaches for imbalanced data streams regression models. Data Min Knowl Discov 35:2389–2466

Bal PR, Kumar S (2018) Cross project software defect prediction using extreme learning machine: An ensemble based study. In: ICSOFT, pp. 354–361

Bal PR, Kumar S (2020) Wr-elm: weighted regularization extreme learning machine for imbalance learning in software fault prediction. IEEE Trans Reliab 69(4):1355–1375

Batista GE, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newslett 6(1):20–29

Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. ACM Comput Surv 49(2):1–50

Branco P, Ribeiro RP, Torgo L (2016) Ubl: an r package for utility-based learning. arXiv preprint arXiv:1604.08079

Branco PO, Torgo L, Ribeiro RP (2017) Smogn: a pre-processing approach for imbalanced regression. In: First International Workshop on Learning with Imbalanced Domains: Theory and Applications, vol. 74, pp. 36–50

Branco P, Torgo L, Ribeiro RP (2018) Rebagg: Resampled bagging for imbalanced regression. In: Second International Workshop on Learning with Imbalanced Domains: Theory and Applications, pp. 67–81

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

Camacho L, Douzas G, Bacao F (2022) Geometric smote for regression. Expert Syst Appl 193:116387

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Cruz RM, Souza MA, Sabourin R, Cavalcanti GD (2019) Dynamic ensemble selection and data preprocessing for multi-class imbalance learning. Int J Pattern Recognit Artif Intell 33(11):1940009

Del Rio S, Benítez JM, Herrera F (2015) Analysis of data preprocessing increasing the oversampling ratio for extremely imbalanced big data classification. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 2, pp. 180–185. IEEE

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Díez-Pastor JF, Rodríguez JJ, García-Osorio CI, Kuncheva LI (2015) Diversity techniques improve the performance of the best imbalance learning ensembles. Inf Sci 325:98–117

Dougherty RL, Edelman AS, Hyman JM (1989) Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation. Math Comput 52(186):471–494

Fritsch FN, Carlson RE (1980) Monotone piecewise cubic interpolation. SIAM J Numer Anal 17(2):238–246

Gado JE, Beckham GT, Payne CM (2020) Improving enzyme optimum temperature prediction with resampling strategies and ensemble learning. J Chem Inf Model 60(8):4098–4107

Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2011) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42(4), 463–484

García V, Sánchez JS, Marqués A, Florencia R, Rivera G (2020) Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. Expert Syst Appl 158:113026

Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239

Hubert M, Vandervieren E (2008) An adjusted boxplot for skewed distributions. Comput Stat Data Anal 52(12):5186–5201

Johnson JM, Khoshgoftaar TM (2019) Survey on deep learning with class imbalance. J Big Data 6(1):1–54

Kovács G (2019) An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Appl Soft Comput 83:105662

Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Prog Artif Intell 5(4):221–232

Kubat M, Matwin S, *et al.* (1997) Addressing the curse of imbalanced training sets: one-sided selection. In: Icml, vol. 97, p. 179. Citeseer

Lee SS (1999) Regularization in skewed binary classification. Comput Stat 14(2):277–292

Lee SS (2000) Noisy replication in skewed binary classification. Comput Stat Data Anal 34(2):165–191

López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250:113–141

Lorena AC, Maciel AI, Miranda PB, Costa IG, Prudêncio RB (2018) Data complexity meta-features for regression problems. Mach Learn 107(1):209–246

Moniz N, Torgo L, Rodrigues F (2014) Resampling approaches to improve news importance prediction. In: International Symposium on Intelligent Data Analysis, pp. 215–226. Springer

Moniz NM, Branco PO, Torgo L (2017) Evaluation of ensemble methods in imbalanced regression tasks. In: Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications, vol. 74, pp. 129–140

Moniz N, Branco P, Torgo L (2017) Resampling strategies for imbalanced time series forecasting. Int J Data Sci Anal 3(3):161–181

Moniz N, Ribeiro R, Cerqueira V, Chawla N (2018) Smoteboost for regression: Improving the prediction of extreme values. In: 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), pp. 150–159. IEEE

Moniz N, Monteiro H (2021) No free lunch in imbalanced learning. Knowl-Based Syst 227:107222

Oliveira Branco PA (2018) Utility-based predictive analytics

Rathore SS, Kumar S (2017) Towards an ensemble based system for predicting the number of software faults. Expert Syst Appl 82:357–382

Rathore SS, Kumar S (2017) Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems. Knowl-Based Syst 119:232–256

Ribeiro R (2011) Utility-based regression. Ph. D. dissertation

Ribeiro RP, Moniz N (2020) Imbalanced regression and extreme value prediction. Mach Learn 109(9):1803–1835

Roy A, Cruz RM, Sabourin R, Cavalcanti GD (2018) A study on combining dynamic selection and data preprocessing for imbalance learning. Neurocomputing 286:179–192

Sáez JA, Krawczyk B, Woźniak M (2016) Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognit 57:164–178

Song XY, Dao N, Branco P (2022) Distsmogn: Distributed smogn for imbalanced regression problems. In: Fourth International Workshop on Learning with Imbalanced Domains: Theory and Applications, pp. 38–52. PMLR

Steininger M, Kobs K, Davidson P, Krause A, Hotho A (2021) Density-based weighting for imbalanced regression. Mach Learn 110:2187–2211

Torgo L, Ribeiro R (2003) Predicting outliers. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 447–458. Springer

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, Ribeiro R (2009) Precision and recall for regression. In: International Conference on Discovery Science, pp. 332–346. Springer

Torgo L, Ribeiro RP, Pfahringer B, Branco P (2013) Smote for regression. In: Portuguese Conference on Artificial Intelligence, pp. 378–389. Springer

Tukey J (1970) Exploratory Data Analysis, limited prelim. ed. Addison-Wesley, Reading, Mass

Wojciechowski S, Wilk S (2017) Difficulty factors and preprocessing in imbalanced data sets: an experimental study on artificial data. Found Comput Decis Sci 42(2):149–176

Yang Y, Zha K, Chen Y, Wang H, Katabi D (2021) Delving into deep imbalanced regression. In: International Conference on Machine Learning, pp. 11842–11851 . PMLR

Zyblewski P, Sabourin R, Woźniak M (2019) Data preprocessing and dynamic ensemble selection for imbalanced data stream classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 367–379. Springer