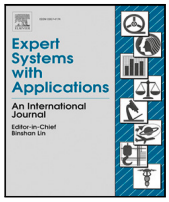
Expert Systems With Applications 252 (2024) 124118

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Oversampling techniques for imbalanced data in regression

Samir Brahim Belhaouari [a], Ashhadul Islam [a], Khelil Kassoul [b,*], Ala Al-Fuqaha [a], Abdesselam Bouzerdoum [a,c]

[a] Division of Information and Computing Technology, Hamad Bin Khalifa University, Qatar
[b] Geneva School of Business Administration, University of Applied Sciences Western Switzerland, HES-SO, 1227 Geneva, Switzerland
[c] School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, Australia

ARTICLE INFO

Keywords:
Data augmentation
Machine learning
AutoInflaters
Nearest neighbor
Imbalanced data

ABSTRACT

Our study addresses the challenge of imbalanced regression data in Machine Learning (ML) by introducing tailored methods for different data structures. We adapt K-Nearest Neighbor Oversampling-Regression (KNNOR-Reg), originally for imbalanced classification, to address imbalanced regression in low population datasets, evolving to KNNOR-Deep Regression (KNNOR-DeepReg) for high-population datasets. For tabular data, we also present the Auto-Inflater neural network, utilizing an exponential loss function for Autoencoders. For image datasets, we employ Multi-Level Autoencoders, consisting of Convolutional and Fully Connected Autoencoders. For such high-dimension data our approach outperforms the Synthetic Minority Oversampling Technique for Regression (SMOTER) algorithm for the IMDB-WIKI and AgeDB image datasets. For tabular data we conducted a comprehensive experiment using various models trained on both augmented and non-augmented datasets, followed by performance comparisons on test data. The outcomes revealed a positive impact of data augmentation, with a success rate of 83.75% for Light Gradient Boosting Method (LightGBM) and 71.57% for the 18 other regressors employed in the study. This success rate is determined by the frequency of instances where models performed better when augmented data was used compared to instances with no augmentation. Access to the comparative code can be found in GitHub.

1. Introduction

The effectiveness of conventional machine learning (ML) techniques greatly depends on the underlying data distribution they are trained on. In scenarios involving classification or regression, a model trained on an imbalanced dataset can exhibit a bias towards the majority class (Gan, Shen, An, Xu, & Liu, 2020; Liu et al., 2018). This may result in seemingly high accuracy, while minority data points are frequently misclassified or mispredicted. Such a scenario can compromise the dependability of ML models, especially in critical domains like healthcare and finance, where rare, malignant, or suspicious data can hold substantial consequences. Recognizing a dataset as imbalanced depends on the specific problem, as illustrated in Fig. 1. This variability highlights the critical need to actively address data imbalance during machine learning model training. Neglecting this aspect can lead to skewed results and compromised model reliability, particularly in critical domains like healthcare and finance, where accurate predictions for rare or suspicious cases are crucial. Thus, acknowledging and mitigating data imbalance stands as a vital step in ensuring the robustness and effectiveness of machine learning applications.

In machine learning, models trained on imbalanced datasets can exhibit a bias towards the majority class, which can impact the reliability of predictions in critical areas such as healthcare and finance. Imbalanced datasets can be found in both classification and regression problems (Fernández et al., 2018). In classification, an imbalanced dataset can have a disproportionate representation of categories, with some categories having fewer samples than others, creating an imbalanced binary or multi-class problem. Binary classification is the primary focus of research in imbalanced learning, but imbalanced data can also arise in regression tasks (Sun, Wong, & Kamel, 2009). In regression, the dependent variable is a continuous value, and the imbalance occurs when a specific interval of the target variable has a reduced representation in the dataset (Branco, Torgo, & Ribeiro, 2016). The imbalanced regression problem is challenging as it requires the model not only to create artificial minority points but also to predict the dependent value for each new data point. Oversampling is a common technique used to address this problem. Fig. 2 illustrates oversampling in imbalanced regression problems, where the target values are used

* Corresponding author.
E-mail addresses: sbelhaouari@hbku.edu.qa (S.B. Belhaouari), aislam@hbku.edu.qa (A. Islam), khelil.kassoul@hesge.ch (K. Kassoul), aalfuqaha@hbku.edu.qa (A. Al-Fuqaha), abouzerdoum@hbku.edu.qa (A. Bouzerdoum).

https://doi.org/10.1016/j.eswa.2024.124118
Received 31 August 2023; Received in revised form 3 March 2024; Accepted 24 April 2024
Available online 20 May 2024

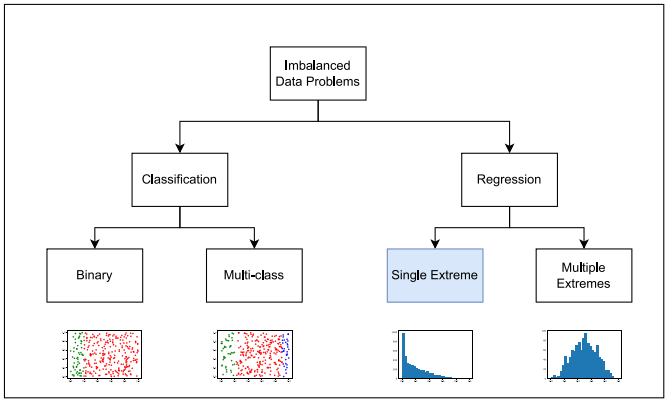


Fig. 1. Types of Imbalance. In this paper we are focusing on the continuous data sets where the target values are skewed to one side.

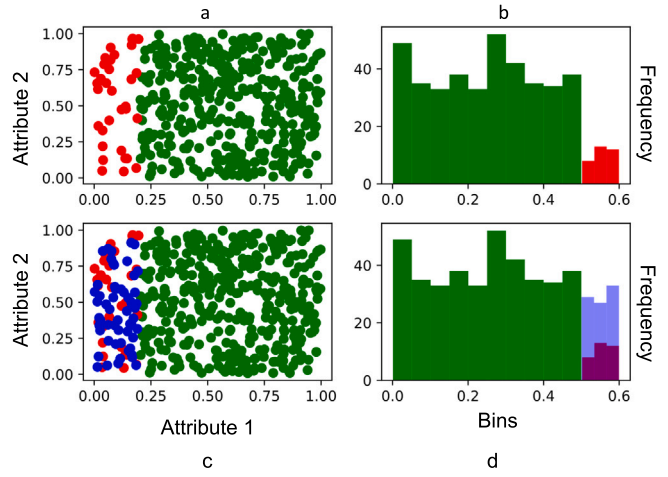


Fig. 2. Oversampling regression on Imbalanced data. a. Imbalanced data set. b. Dependent value distribution. The values in green correspond to the majority values, while the red bars represent the frequency of the values on the minority dataset. The process of augmentation has two parts. c Adding more data points to the minority set. d. Setting target values corresponding to the newly created features. The blue section of the histogram in d shows the increased frequency of values between 0.5 and 0.6 due to the creation of new data points and corresponding target values.

to identify minority data points. The imbalanced regression dataset is visually depicted in Fig. 2a, where the majority and minority data points are represented in green and red, respectively. A histogram in Fig. 2b displays the distribution of dependent values, where the green bars represent the majority values, and the red bars indicate the frequency of minority values. Typically, the identification of imbalanced regression data begins with examining the target values and labeling the corresponding data points as minority.

In Fig. 2b, the histogram illustrates that target values falling within the range of 0 to 0.5 (highlighted in green) exhibit a higher frequency compared to the range from 0.5 to 0.6 (highlighted in red), which displays a lower frequency. Fig. 2a represents the independent variables, primarily constituting the majority dataset depicted in green, whereas the minority dataset is represented by red scatter points in Fig. 2a. Notably, these red scatter points correspond to the red segment of the histogram in Fig. 2b. The process of augmentation yields two key outcomes. The first outcome involves generating additional data points that resemble those in the minority dataset, symbolized by the blue points in Fig. 2c. The second outcome encompasses the target values linked to these newly generated data points, as depicted in the blue section of the histogram in Fig. 2d. This augmentation process serves to augment the frequency of values falling within the 0.5 to 0.6 range.

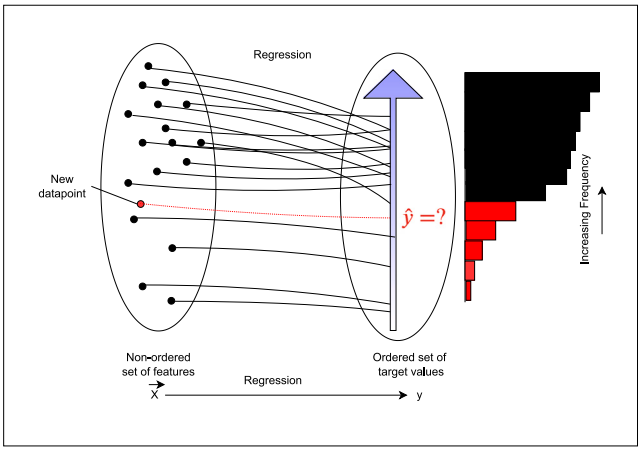


Fig. 3. Domain and Co-Domain relationship in imbalanced regression. The non-ordered set of features is represented on the left. The right side shows the histogram of the target values, increasing vertically upwards. As the regression relationship between the features is unknown, we need first to generate new examples of features, ensuring that it represents the minority data set. Then we compute the possible target value corresponding to the newly created feature.

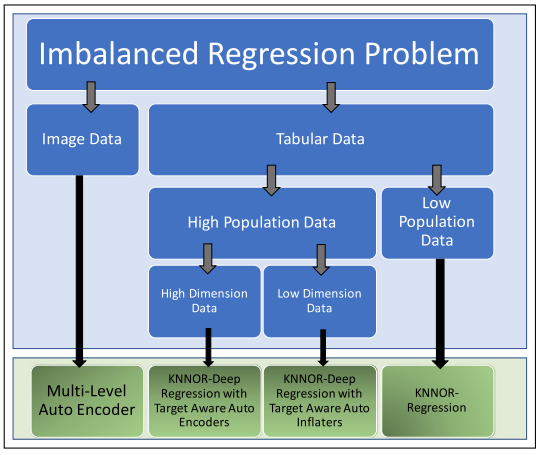


Fig. 4. Different approaches presented in this paper according to the type of dataset under scrutiny. Regarding Tabular data, we offer different methods depending on the population and number of features. We propose KNNOR-Regression (Section 3.2) for low population data. For data with a high population, we advocate KNNOR Deep Regression which has two flavors. For high-population data with a high number of features, we use Target Aware Autoencoders (Section 3.3.1), and for high-population data with a low number of features, we use AutoInflaters (Section 3.3.2). Finally, for Image datasets, we use a combination of Convolution and Fully Connected Autoencoders called Multi-Level Autoencoders (Section 3.4). We define high population data in terms of its volume wherein the entire data cannot be stored in a single machine (Juez-Gil, Arnaiz-González, Rodríguez, & García-Osorio, 2021).

The process is further explained in Fig. 3, where a domain-range relationship is depicted. The left side shows the non-ordered set of features $\hat{X}$, while the target values y are represented on the right in increasing order of their frequencies. The research question is to generate a new representation of $\hat{X}$ that increases the representation of the minority dataset, and then map the new X-value to a y-value that falls within the range of lower frequencies. This requires an approximation of the regression algorithm only for the minority or rare data, achieved by creating specialized functions using simple statistical methods as well as deep neural networks, which are elaborated on in the following pages.

The imbalanced regression problem is apparent in many real-world tasks like medical applications where the different health metrics like blood pressure, heart rate, and Oxygen saturation are continuous,

and their distribution is often skewed across the patient population. Other industries like finance, meteorology, and fault diagnosis are also plagued with imbalanced regression problems (Krawczyk, 2016). This article proposes several novel methods of oversampling Imbalanced Regression Data. The advantages of the proposed techniques in this study, when compared to those identified in the existing literature, can be summarized as follows: Firstly, we extend the K Nearest Neighbor Oversampling (KNNOR) method (Islam, Belhaouari, Rehman, & Bensmail, 2022b) to KNNOR-Regression (KNNOR-Reg), enabling the generation of target values for imbalanced regression problems. Additionally, we expand upon Class Aware Autoencoders by introducing Target Aware Auto Encoders, which are designed for estimating target values for new features. We also introduce a novel architecture known as Target Aware AutoInflaters, serving to extract features from low-dimensional data. Furthermore, our study involves the development of Multi-level Auto Encoders, which are adept at extracting features from images and generating new features and target values. To enhance prediction accuracy, we employ an exponential loss function within the AutoInflater, effectively highlighting the differences between predicted and actual target values. Lastly, our approach incorporates the use of the maximum test target value as a normalizer for calculating regression loss, providing a comprehensive and effective framework for addressing the identified challenges.

The contributions of this paper are summarized as below.

- Extending the KNNOR method to KNNOR-Regression (KNNOR-Reg) to generate target values for imbalanced regression problems (see Section 3.2);
- Extending Class Aware Autoencoders to Target Aware Auto Encoders for estimating target values for new features (see Section 3.3.1);
- Introducing a novel architecture called Target Aware AutoInflaters to extract features from low dimensional data (see Section 3.3);
- Developing Multi-level Auto Encoders for extracting features from images and creating new features and target values (see Section 3.4);
- Using an exponential loss function within the AutoInflater to better highlight the difference between predicted and actual target values (see Section 3.3.3);
- Calculating regression loss using the maximum test target value as a normalizer (see Section 4.1.1).

The paper is structured as follows: In Section 2, we introduce the issue of imbalanced linear regression, explain its significance, and provide a review of existing literature on the subject. In Section 3, we present our solutions to address this problem, which are categorized into two frameworks based on dataset size. For smaller datasets, we introduce an extended version of KNNOR, while for larger datasets, we propose a novel AutoEncoders implementation. Fig. 4 offers an overview of our proposed methods, organized by data type and structure. For tabular data, we propose different methods, such as KNNOR Regression (Section 3.2) and KNNOR DeepRegression with target-aware Auto Encoders/Inflaters (Sections 3.3.1 and 3.3.2), depending on dataset size and feature count. In the case of image datasets, we suggest a Multi-Level AutoEncoder scheme (Section 3.4). In Section 4, we outline the experimental design, where we assess the effectiveness of our methods on well-known imbalanced regression datasets and present the results and subsequent discussion. Section 5 encompasses the conclusion and outlines future work based on the results.

2. Presentation of the problem and related work

In this section, we begin by introducing the concept of imbalanced linear regression, elucidating its importance, and conducting a comprehensive examination of the existing body of literature pertaining to this topic.

2.1. Problem description

In a standard regression problem, the goal is to predict continuous values based on a set of examples and their corresponding target values. However, in the case of an imbalanced regression problem, the data distribution may be skewed, with only a few examples having target values within a specific range of interest, while the majority of examples have target values in a different range. This imbalance can result in a biased model that tends to predict values within the range of the majority data. This bias occurs because most regression algorithms are designed to minimize the average error across all data points (Gan et al., 2020; Liu et al., 2018). As a consequence, the model's accuracy is typically higher for the majority class but lower for the minority class. This can be misleading when assessing the overall model accuracy. In fact, a classifier or regressor may predict the entire dataset to belong to the majority class or within the majority range, thus getting the minority class or rare data wrong, while still achieving a seemingly high accuracy due to the averaging effect. This phenomenon is important to consider when evaluating the performance of models in imbalanced regression scenarios (Fernández et al., 2018; Gan et al., 2020; Liu et al., 2018).

In the context of imbalanced regression, it is necessary to define the relevant terms. Let $D = \left\{\left(\mathbf{x}_i, y_i\right)\right\}_{i=1}^{N}$ denote the set of training data, where $\mathbf{x}_i \in \mathbb{R}^d$ represents the input features and $y_i \in \mathbb{R}$ represents the dependent value, which is continuous in nature. To further characterize the dependent value space, Branco, Torgo, and Ribeiro (2019) introduce a threshold value t_r that divides the dataset into two complementary sets: the common data, represented by D_N, and the rare data, denoted by D_R, where $|y_i| < t_r$ indicates rare data and $|y_i| \geq t_r$ indicates common data. The imbalanced regression problem arises when the following conditions hold:

- Accurate prediction of D_R is more crucial for determining the performance of the model;
- $D_R \ll D_N$, where D_R and D_N represent the cardinalities of the rare and common datasets, respectively.

Minority and Majority in Continuous data

In contrast to classification problems, labeling in regression problems can be more complex since the focus is on identifying rare events or valuable data points, such as fraudulent transactions, highly profitable stock market actions, or ecological catastrophes. Therefore, the identification of minority data points is of utmost importance. However, it is also essential to consider that misclassifications can have different costs. To address this, the utility theory is used to define a relevance function that assigns importance to each target value (Torgo & Ribeiro, 2007). The relevance function is a continuous, real-valued function that is dependent on the domain and maps each target value to a relevance scale. Eq. (1) defines a relevance function that takes into account the application-specific bias and maps each target value to a continuous scale of relevance ranging from 0 to 1, where 0 indicates minimum and 1 indicates maximum importance.

$$\phi(Y) : \mathcal{Y} \rightarrow [0, 1] \tag{1}$$

To obtain the relevance function, we use the box and whisker plot of the target value, where the median value is assigned an importance value of 0 and the upper adjacent and all higher values are assigned an importance value of 1. Similarly, all lower adjacent values are assigned an importance value of 1. To interpolate between these importance values and obtain a smooth relevance function, we use a piece-wise cubic Hermite interpolation method (Camacho, Douzas, & Bacao, 2022).

The relevance values, calculated using the same method, are illustrated in Fig. 5. In Fig. 5a, the histogram represents the target values of the compactiv dataset, with the corresponding relevance values depicted below. Notably, there is a correlation between the relevance values and the frequency of the target values. The gap in the histogram is responsible for the discontinuity in the plot. Moving to Fig. 5b, it

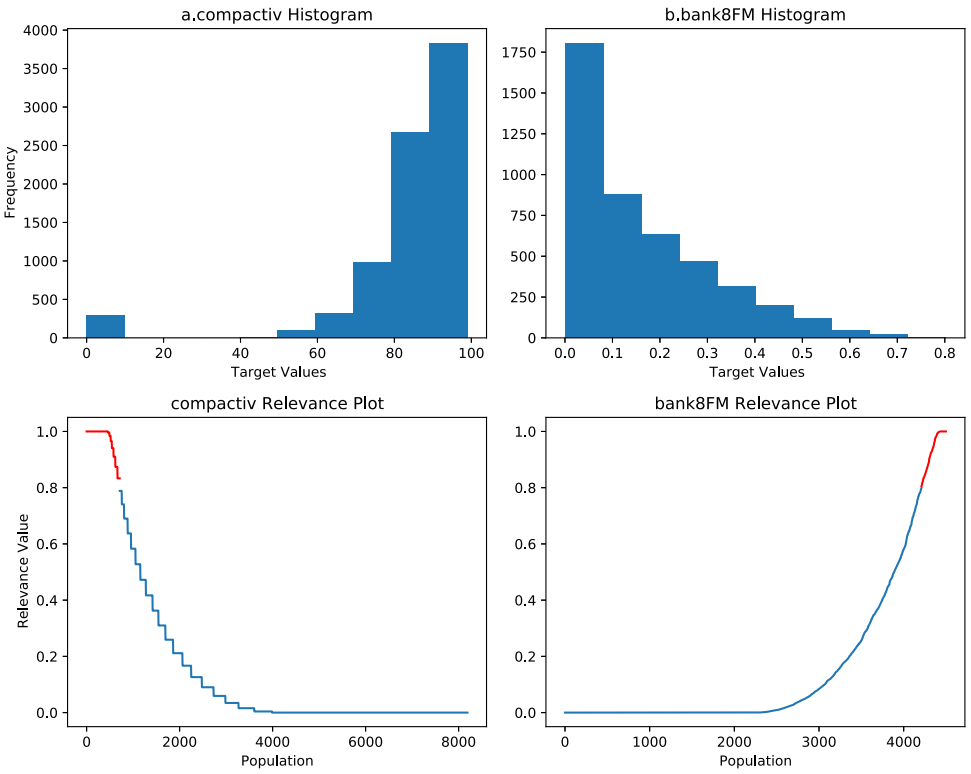


Fig. 5. Relevance function for the a. compactiv and b.bank8FM data set. Figure a displays a histogram depicting the distribution of target values within the compactiv dataset, with the corresponding relevance values presented below. It is worth noting that there is a noticeable correlation between these relevance values and the frequency of target values. The interruptions in the plot can be attributed to gaps within the histogram. Figure b shows the histogram for the bank8FM dataset, along with the associated relevance values displayed below it.

shows the histogram for the bank8FM dataset, along with the associated relevance values displayed beneath it. It is worth mentioning that the positioning of the extremes is a crucial factor in determining the relevance function. In this exercise, we considered datasets with either high (Fig. 5b) or low (Fig. 5a) extremes but not both. The red and blue sections in the lower half of the figure are also of significance. The threshold of importance is manually defined, with target values above this threshold considered critical. We adhere to established practices (Camacho et al., 2022; Torgo & Ribeiro, 2007) to partition the data into two subsets: D_N and D_R. For each dataset, a user-defined coefficient is employed to determine the extent to which the whiskers extend from the interquartile range in the box plot of the dependent data. This threshold plays a pivotal role in segregating the data into rare (D_R) and common (D_N) subsets. We rely on the methodology proposed by Camacho et al. (2022) to derive these thresholds. Once the data has been categorized into these two segments, our algorithm can be applied. However, before delving into the specifics of our proposed methods, it is essential to review recent work in the field of imbalanced regression.

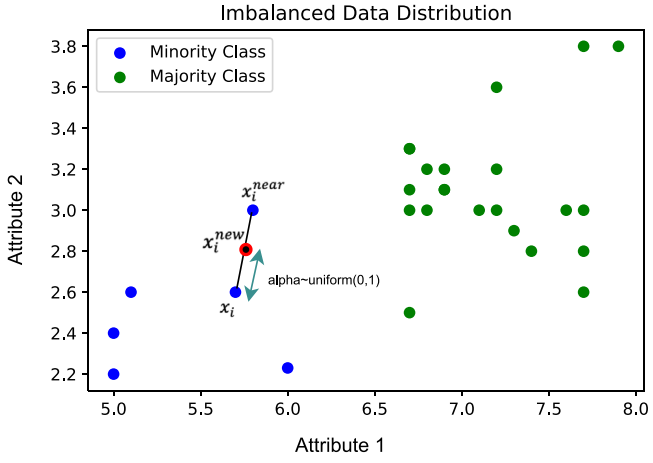


Fig. 6. SMOTE — the fundamental augmentation algorithm. x_i^{new} is the generated point at a random distance between x_i and $x_i^{nearest}$.

2.2. Literature review

In a comprehensive exploration of machine learning applications, particular emphasis is placed on addressing imbalanced data augmentation challenges. One study investigates the use of augmentation techniques to balance data, especially in the context of carbon oxides (CO) and nitrogen oxides (NOx) emissions prediction from a gas turbine (dos Santos Coelho, Hultmann Ayala, & Cocco Mariani, 2024). The research unveils the importance of hyperparameter tuning and feature engineering, particularly with the Deep Forest Regression (DFR) model, in enhancing predictive accuracy for these emissions. Additionally, the study delves into the small punch test (SPT) and employs machine learning to establish the correlation between SPT forces and material properties, offering new insights into predicting material tensile properties using SPT (Zhong, He, Guan, & Jiang, 2023). In a soil-related investigation, machine learning algorithms are harnessed to estimate soil properties, with artificial neural networks (ANN) emerging as the most effective predictor (Tunçay, Alaboz, Dengiz, & Başkan, 2023). These studies collectively underscore the significance of machine learning in addressing imbalanced data challenges and optimizing predictive capabilities across diverse domains.

In the realm of addressing imbalanced data in classification, particularly in the context of oversampling, there are primarily three fundamental approaches. These approaches are aimed at mitigating imbalanced data issues, particularly when dealing with classification

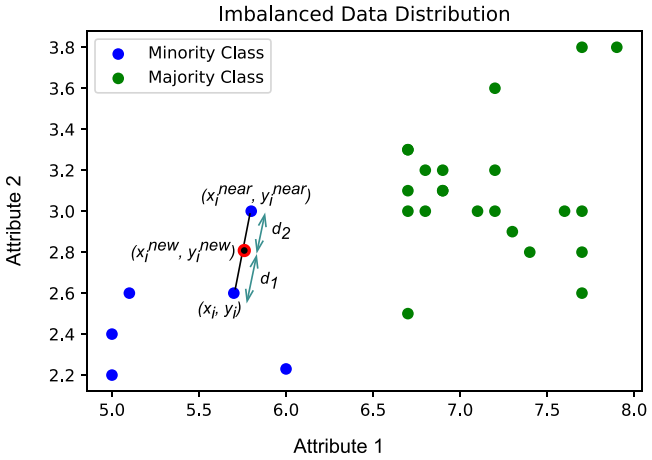


Fig. 7. Application of SMOTER to generate new point as well as target value.

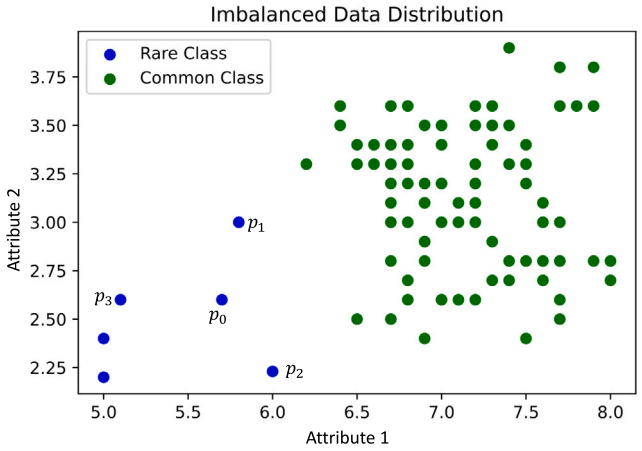


Fig. 8. Simulated dataset — before augmentation. p_0 is the source point from where augmentation will start. p_1, p_2 and p_3 are its 3 nearest neighbors in increasing order of distance.

problems: (1) Data-Driven Techniques, as described by Laza, Pavón, Reboiro-Jato, and Fdez-Riverola (2011), involves sampling-based methods that focus on adjusting the distribution of each category within the dataset. Data-driven techniques are more universally applicable and widely employed due to their adaptability and effectiveness; (2) Algorithm-Based Methods, as exemplified by Elhassan and Aljurf (2016), Kubat et al. (1997), Thanathamathee and Lursinsap (2013), entail modifications to the training algorithms used in the classification process. These adjustments are specific to the classifiers employed and may not be as common as data-driven techniques. However, they can prove to be highly effective in specific cases; (3) Hybrid Approaches, as proposed by Johnson and Khoshgoftaar (2019), combines elements of both data-driven and algorithm-based techniques. This approach seeks to leverage the advantages of each method and is gaining attention for its potential to deliver comprehensive solutions to imbalanced data challenges.

The choice of which approach to employ depends on the specific characteristics of the dataset and the problem at hand. Data-driven techniques are often favored for their broad applicability, but algorithm-based methods and hybrid approaches can be valuable in situations where tailored adjustments to classification algorithms are required. Data-driven techniques for handling imbalanced classification problems often use oversampling or undersampling methods (He & Garcia, 2009; Liu, Wu, & Zhou, 2008). A more intelligent approach is Synthetic Minority Oversampling Technique (SMOTE) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), which creates new data points belonging to the minority class by using existing minority data. As illustrated in Fig. 6, this algorithm selects a random minority point and places a new one at a random distance between the chosen point and one of its closest neighbors. Mathematically, SMOTE generates an artificial point x_i^{new} according to Eq. (2):

$$x_i^{new} = x_i + (x_i - x_i^{nearest}) * \alpha_i \tag{2}$$

where x_i is a minority data point, its nearest neighbor of same class is $x_i^{nearest}$ and α_i is independent and identically distributed number uniformly distributed on [0, 1].

2.2.1. SMOTE based imbalanced regression

A fundamental solution to address imbalanced regression is the use of random under-sampling and Synthetic Minority Oversampling Technique for Regression (SMOTER) (Torgo, Branco, Ribeiro, & Pfahringer, 2015). Under-sampling involves randomly selecting data points with relevance values below a specified threshold and removing them from the dataset. SMOTER is an extension of SMOTE (Chawla et al., 2002) that generates new data points for the minority class by using existing data. As shown in Fig. 6, the algorithm first selects a random minority point and then places a new point at a random distance between the chosen point and one of its closest neighbors. SMOTER then calculates the possible target value for the new point based on its distance to its parent points, as illustrated in Fig. 7. The new target value y_i^{new} is computed using Eq. (3):

$$y_i^{new} = \frac{\frac{y_i}{d_1} + \frac{y_i^{nearest}}{d_2}}{\frac{1}{d_1} + \frac{1}{d_2}} \tag{3}$$

where, y_i is the target value of the x_i data point. Its nearest neighbor is $x_{nearest}$ whose target value is $y_i^{nearest}$. The target value of the generated point x_i^{new} is proportional to the euclidean distance of the new point from the parent points. Branco et al. (2019) propose three approaches to address imbalanced datasets: random oversampling, the introduction of Gaussian Noise, and WERCS. Random oversampling involves creating copies of the rare class data points to add balance to the dataset. Gaussian Noise is used to introduce small variations in the feature set and target values to generate new data points with dependent values. The WERCS technique combines over and undersampling by assigning a probability of duplication or deletion based on the relevance value of each data point, determined by a user-defined threshold. More recently, the Geometric SMOTE (G-SMOTE) (Douzas & Bacao, 2019) approach has been used in regression (Camacho et al., 2022). In G-SMOTE, data points are classified as rare or common based on the target value, and new data points are generated using the G-SMOTE method. The label of the new data point is the weighted average of the target values of the two instances used to create the new point (Camacho et al., 2022). In this paper, we use a recently published approach that enhances SMOTE with the KNNOR approach (Islam et al., 2022b). The KNNOR algorithm combines SMOTE with KNN to generate new synthetic data points and addresses the issue of the oversampling of noisy and ambiguous instances.

2.2.2. K Nearest Neighbor OveRsampling approach (KNNOR)

The KNNOR has been proposed as a solution to the challenge of class imbalance in classification tasks. Compared to the popular SMOTE algorithm, KNNOR addresses issues such as noisy data, small disjuncts, and within-class imbalances, as demonstrated by Islam and Belhaouari (2021). One of the key features of KNNOR is its novel filtering method, which helps to identify minority data points that better represent the population.

To generate new synthetic data points, KNNOR uses multiple nearest neighbor points of these crucial minority points. The process of creating an artificial data point begins by selecting one of the crucial

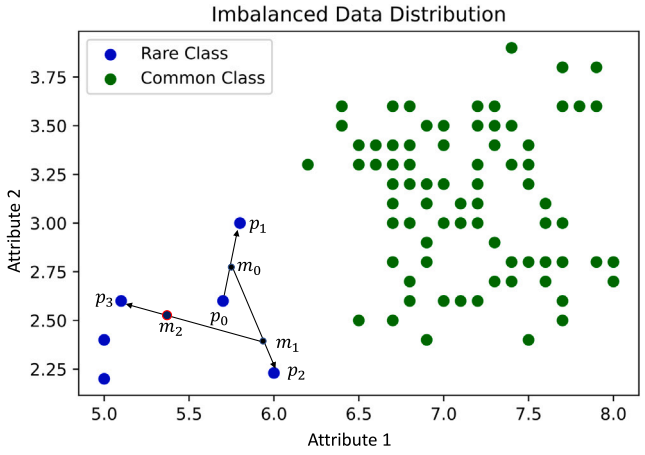


Fig. 9. The iterative process of creation of an augmented point m_2 with the help of three neighbors (p_1, p_2 and p_3), starting with p_0.

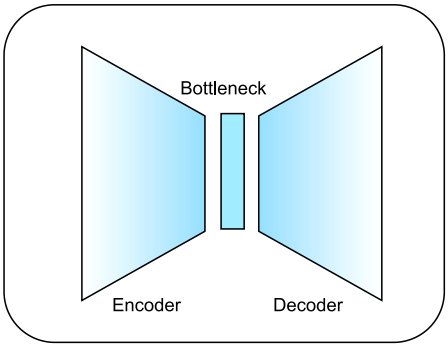


Fig. 10. Auto Encoder Block Diagram.

minority points, denoted as x_0, and repeating the following steps for each of its k nearest neighbors:

- generate a random point on a line between the start point and the next closest neighbor;
- make the point generated as the start point.

Fig. 8 shows an artificial imbalanced dataset and Fig. 9 gives a pictorial representation of the process of augmentation using three neighboring points. In a general case where k neighbors are used to create an artificial point, the process can be represented by the following.

$\forall i \in [0, 1...k]$ the sequence is defined using Eq. (4):

$$x_{i+1}^{new} = x_i^{new} + (p_i - x_i^{new}) * \alpha_i \tag{4}$$

where x_0^{new} is any safe point in the minority class, p_i is the ith nearest neighbor of x_0^{new} and α_i is uniform random variables over $[0, M]$, where M is any positive value less than 1.

At each iteration of the process, the new point generated at the preceding step becomes the starting point. A new point is synthesized at a distance $random(0, M)$ on the straight line joining the starting point and the $(i+1)^{th}$ nearest neighbor of the origin point that started the exercise. The synthetic point obtained after the last iteration is considered the new augmented data point for the entire process.

2.2.3. Deep learning in imbalanced regression

In addition to statistical techniques, this paper proposes a deep learning approach, which is crucial to understanding imbalanced regression using deep neural networks. Neural networks are particularly useful for high-dimensional datasets, such as images, and have been successfully applied in various areas, such as age detection from facial images, weather prediction, electricity consumption estimation, and autonomous vehicle trajectory projection, where rare data is present at the extremes. Recent research in imbalanced regression using deep neural networks has resulted in notable works, such as Label and Feature Distribution Smoothing (LDS and FDS, respectively) (Yang, Zha, Chen, Wang, & Katabi, 2021), where different kernel functions are applied to the labels and features to create a more balanced data distribution. To improve performance, the loss functions are re-weighted by multiplying them with the inverse of the estimated LDS. A recent study by Sharan et al. (2023) focuses on the domain of probabilistic forecasting, addressing the challenge of predicting long-tailed rare data, as discussed by Menon (2020). The authors of this study introduce novel concepts related to moment-based tailedness measurements to improve predictions. They propose two loss functions: the Kurtosis loss, which assesses the fourth moment around the distribution mean and is symmetric, and the Pareto loss, which evaluates the right-tailedness of the distribution and is asymmetric. Notably, this paper stands out for its innovative approach, as it combines deep learning and statistical methods to create artificial samples with precise target values. To achieve this, the authors extend the capabilities of a specialized neural network model known as AutoEncoders, which is employed to extract relevant features and generate accurate target values.

2.2.4. AutoEncoders

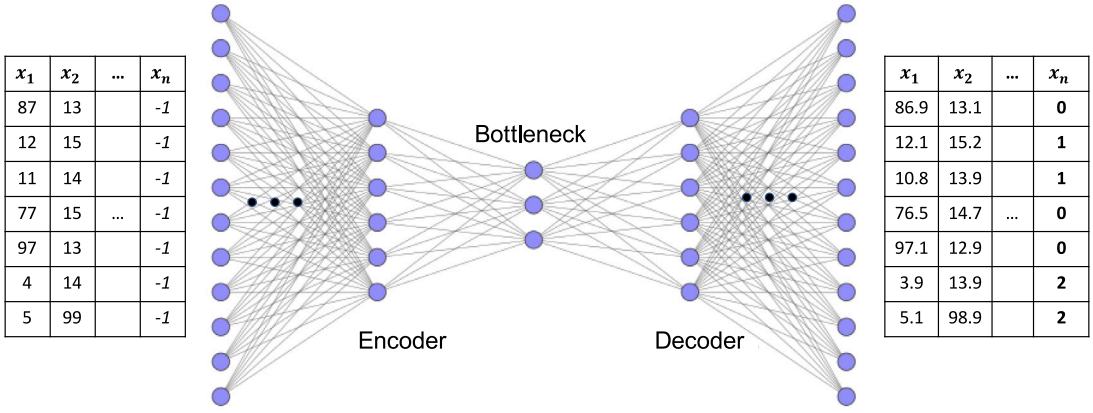
Autoencoder neural networks have the ability to generate output features that match the input features. They are composed of three main components: the encoder, the bottleneck, and the decoder (Rifai, Vincent, Muller, Glorot, & Bengio, 2011). Due to their common usage in image data, autoencoders typically have high-dimensional input features that are reduced to the bottleneck size by the encoder. The decoder is then trained to reconstruct the initial output by minimizing a cost function. Fig. 10 provides a block diagram of an autoencoder that includes the Encoder, Bottleneck, and Decoder. Autoencoders can be fully connected or can include Convolution and De-convolution layers (Zeiler, Krishnan, Taylor, & Fergus, 2010), with the bottleneck typically being a fully connected layer that extracts a one-dimensional feature representation. Autoencoders are used to reduce high-dimensional datasets like images, making them more suitable for statistical methods (Wang, Yao, & Zhao, 2016). This work employs an innovative form of autoencoder known as the Class-Aware Autoencoder (Islam, Belhaouari, Rehman, & Bensmail, 2022a), which is further explained below.

Class Aware AutoEncoders

Autoencoders aim to minimize the difference between input and output data. However, class-aware autoencoders take this a step further by incorporating the class label information into the output data. This means that the output of the class-aware Autoencoder includes both a close approximation of the input feature set and the corresponding class label (Islam et al., 2022a). Fig. 11 illustrates the concept of a class-aware Autoencoder. To match the dimensions, a random or constant feature is added to the input data, and the output is then matched with the actual class label. This approach has been primarily used in labeled datasets for classification tasks. However, it can also be extended to regression data and applied to predict the target value for new data points by modifying the loss function, as described in Section 3.3.1.

3. Material and methods

In this section, we outline our strategies for addressing this issue, which are classified into two frameworks based on dataset size. For smaller datasets, we introduce an extended version of KNNOR, while for larger datasets, we propose a novel AutoEncoders implementation. Fig. 4 provides an overview of our proposed methods, categorized by data type and structure. Regarding tabular data, we propose various methods, including KNNOR Regression (Section 3.2)



x_1	x_2	...	x_n
87	13		-1
12	15		-1
11	14		-1
77	15	...	-1
97	13		-1
4	14		-1
5	99		-1

x_1	x_2	...	x_n
86.9	13.1		**0**
12.1	15.2		**1**
10.8	13.9		**1**
76.5	14.7	...	**0**
97.1	12.9		**0**
3.9	13.9		**2**
5.1	98.9		**2**

Fig. 11. Class Aware Auto Encoder.

and KNNOR DeepRegression with target-aware AutoEncoders/Inflaters (Sections 3.3.1 and 3.3.2), depending on dataset size and feature count. For image datasets, we recommend a Multi-Level AutoEncoder scheme (Section 3.4).

3.1. Methods

This paper introduces three innovative techniques for creating target variables that cater to the structure and size of the dataset.

- Method 1 (For low population data): We propose the KNNOR approach with an additional step of calculating the target variable, which we refer to as KNNOR-Regression (or KNNOR-Reg) in Section 3.2.
- Method 2 (For high-population data): We present KNNOR Deep Regression or KNNOR-DeepReg, which has two variations:
  - Method 2a (For high-population and high-dimensional data): Combination of KNNOR with Target Aware AutoEncoder (Section 3.3.1).
  - Method 2b (For high-population and low-dimensional data): Combination of KNNOR with Target Aware AutoInflater (Section 3.3.2).
- Method 3 (For image data): We propose a Multi-Level AutoEncoder scheme for imbalanced image regression problems, which we discuss in Section 3.4.

3.2. KNNOR - Regression (KNNOR-Reg) - [Low population data]

The approach described in 2.2.2 expands on the KNNOR approach. While KNNOR is primarily used for classification data, generating labels for the artificial minority data point is a straightforward task in that context. However, when dealing with regression, we maintain a record of each point involved in producing the new point. After creating the new point, we consider the distance between the artificial point and each of these points to determine the target value. The distance, denoted as $d\left(y_j, y_i\right) = ||x_j - x_i||$ and represented as d_j, can be generalized as follows. If d_i is the distance of the artificial point x_{new} to the i_{th} parent point x_i, and y_i is the target value for this point, then the target value y_{new} corresponding to the freshly created data point is expressed using Eq. (5) as follows:

$$y_i^{\text{new}} = \sqrt[\alpha]{\frac{\sum_{j=1}^{k}\left(\frac{y_j}{d(x_j, x_i^{\text{new}})}\right)^{\alpha}}{\sum_{j=1}^{k}\left(\frac{1}{d(x_j, x_i^{\text{new}})}\right)^{\alpha}}} \tag{5}$$

where $y_i = Reg(x_i)$, Reg() is the function that we try to estimate better. The value $d(y_j, y_i) = ||x_j - x_i|| = d_j$ and x_j is the nearest neighbor to x_i. The vectors x_1 to x_k are the k nearest neighbors of x_i^{new} The value of α is 2 in case of euclidean distance or infinity in case of higher dimension. In our experiments we have used the value of α as 2. The number of points or neighbors that participated in the creation of the point is denoted by k. When the value of α is 1 and k is 2, it gives the original case of SMOTE-Regression. When the value of k changes from 2 onward, it is an application of the KNNOR-Regression method. The

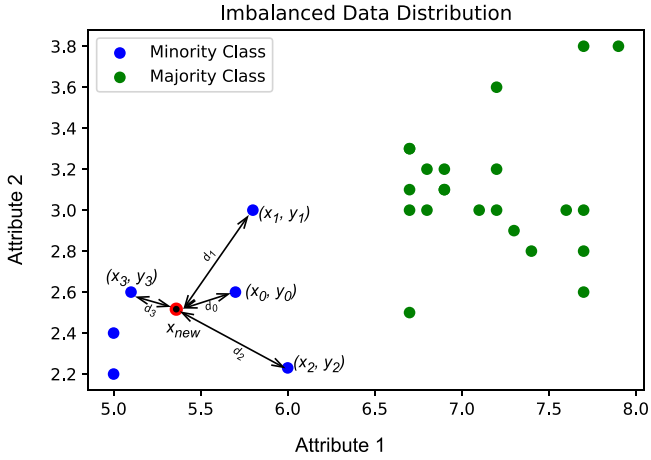


Fig. 12. The extension of KNNOR to estimate the target value using the distances of the new point x_{new} to the points associated with its creation — starting with x_0 followed by x_1, x_2 and x_3.

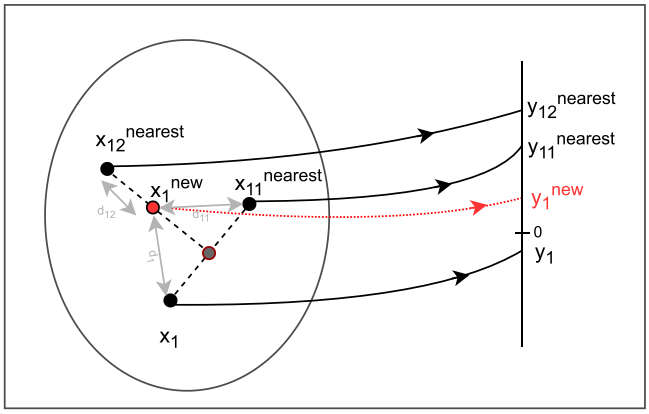


Fig. 13. The extension of KNNOR to estimate the target value using the origin point x_1 and two of its nearest neighbors, $x_{11}^{nearest}$ and $x_{12}^{nearest}$.

process is illustrated in Fig. 12 which uses a weighted sum of the target values corresponding to x_0, x_1, x_2 and x_3 to generate the target value for the new data point.

An alternative description of the process is shown in Fig. 13. The origin point is x_1 and two of its nearest neighbors are $x_{11}^{nearest}$ and $x_{12}^{nearest}$. x_1^{new} is the artificial point created by using x_1 and its two nearest neighbors following the process of KNNOR (Islam et al., 2022b). While the new point x_{new} is being generated, we keep track of the points and neighbors involved in the operation. In this case, they are x_1, $x_{11}^{nearest}$ and $x_{12}^{nearest}$. Their corresponding target values are represented in the vertical line on the right side of Fig. 13. The target value corresponding to x_1 is y_1, $x_{11}^{nearest}$ is $y_{11}^{nearest}$ and $x_{12}^{nearest}$ is $y_{12}^{nearest}$. We also measure the distance between the newly created point x_1^{new} and the three points x_1, $x_{11}^{nearest}$ and $x_{12}^{nearest}$. The distances are d_1, d_{11} and d_{12} respectively. Finally, we utilize Eq. (5) in the following manner, as specified in Eq. (6) as follows:

$$y_1^{new} = \frac{\frac{y_1}{d_1} + \frac{y_{11}^{nearest}}{d_{11}} + \frac{y_{12}^{nearest}}{d_{12}}}{\frac{1}{d_1} + \frac{1}{d_{11}} + \frac{1}{d_{12}}} \tag{6}$$

where α is set to 1. The process of generating artificial data points and their corresponding target values using a combination of KNNOR and SMOTE-Regression is outlined in Algorithm 1.

Algorithm 1 KNNOR Regression (KNNOR-Reg)

Input Training data set S_{train};
number of neighbors k;
count of datapoints to be augmented aug_num;
Output Artificial point with target values
new_points = generate artificial points using KNNOR
$new\_targets$ = empty array of size(new_points)
For point p in each new_points do
  k_nbrs = the k points that were used to create p
  $weighted\_sum\_distance = 0$
  $weights = 0$
  For each neighbor n in k_nbrs do
    $dist$ = distance between n and p
    $weighted\_sum\_distance$ += (target value of n)/$dist$
    $weights$ += 1/$dist$
  End For
  $final\_target\_value = weighted\_sum\_distance/weights$
  add $final\_target\_value$ to $new\_targets$
Return new_points, $new\_targets$

The process of generating artificial data points along with their corresponding target values is elucidated through a flowchart, as depicted in Figs. 14 and 15. Fig. 14 outlines the methodology for creating new data points, while Fig. 15 illustrates the procedure for computing target values associated with the newly generated data points.

3.3. KNNOR-Deep Regression (KNNOR-DeepReg) — [High population data]

Although KNNOR-Reg is a powerful approach for data imputation and the inclusion of multiple neighbors introduces non-linearity, it is possible to enhance the method further by utilizing neural networks. This concept draws inspiration from our previous research on Class Aware Autoencoders (Islam et al., 2022a), as described in Section 2.2.4. In this study, we expand upon Class Aware Autoencoders and introduce Target Aware Autoencoders and Target Aware AutoInflaters, as outlined below.

3.3.1. Target Aware AutoEncoders - [High population, high-dimension data]

To delve deeper into the concept of Target Aware Autoencoders, we refer to Fig. 16, which represents our objective. Our aim is to train a neural network that can predict target values while simultaneously

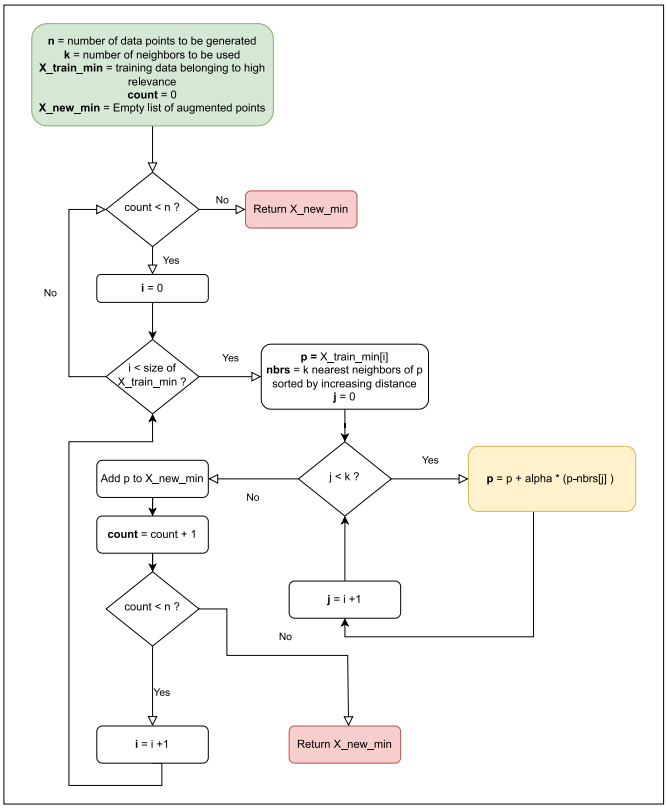


Fig. 14. Flowchart showing the process of generation of novel artificial data points using the k nearest neighbors.

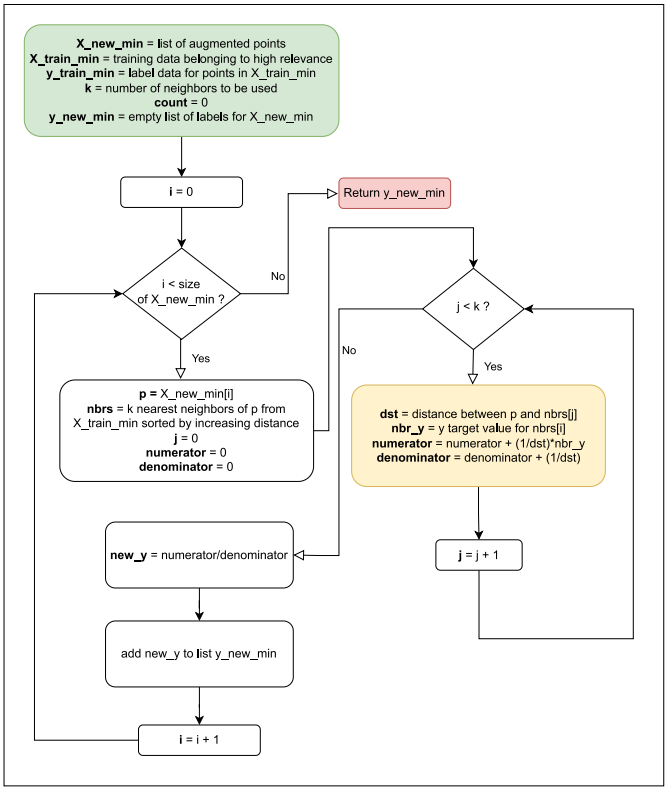


Fig. 15. After generation of the new data points (Fig. 14), this flowchart shows the process of computing the corresponding target values of each novel data point by using the target values of the k nearest neighbors.

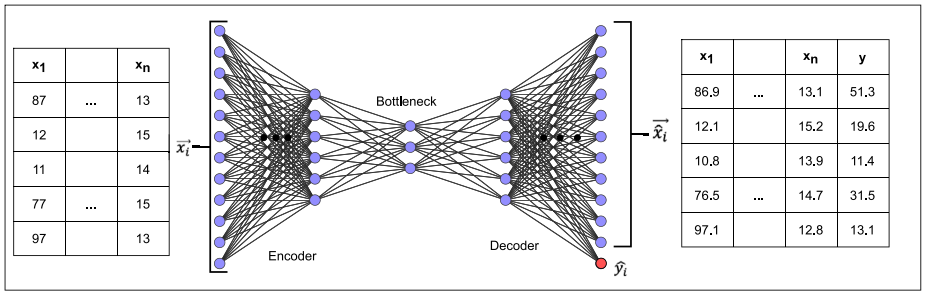


Fig. 16. Target Aware Auto Encoder (Fully Connected). The neuron highlighted in red represents the additional value generated, matching the target value. The network is capable not only to extract features but also estimate the target value. The latter is generated by including an additional component in the loss function.

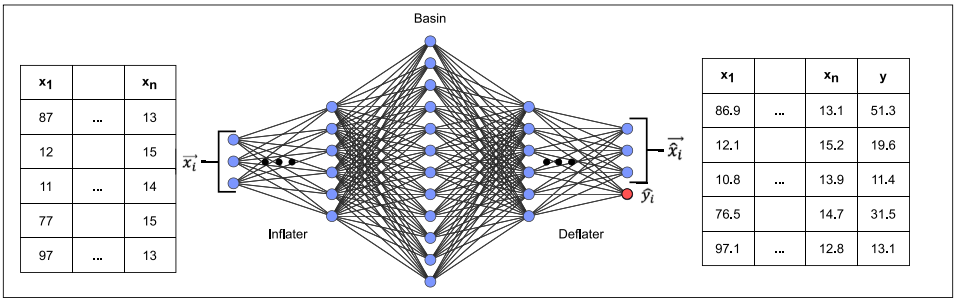


Fig. 17. Target Aware Inflater-Deflater (Fully Connected) for low dimension data. The neuron highlighted in red represents the additional value generated, matching the target value. The bottleneck expands the feature set while the Network gives an additional output, the target value corresponding to the features. This is done by including the target value in the output as well as the loss function.

learning the features of the rare dataset. In this context, the Autoencoder is defined as the function F, from $\mathbb{R}^d$ to $\mathbb{R}^{d+1}$ using Eq. (7) as follows:

$$F(X) = (\hat{X}, \hat{y}) \tag{7}$$

where X is the input feature, $\hat{X}$ and $\hat{y}$ are the approximations of input features and target values by the model. The model can thus mimic the input features on the output side and also generates a target value that it learns while learning the features. Thus, although on the input side, we only provide the features, on the output, we enforce the target-aware model to generate an estimated target value. This is achieved by defining a new loss function that enables the model to learn from the features as well as the targets. It is explained in the following Section 3.3.3. Initially, the model has a high error in generating the target value, gradually improving with multiple iterations. It is interesting to note here that the target values, y are not represented in the input of the neural network unlike the class aware auto encoders (Islam et al., 2022a) as tests have shown that removing the target from the input achieves better results. The second loss term is fed to the model externally during every training batch to gauge the features and the target. The process is shown in Fig. 16 where the neuron highlighted in red captures the additional value generated, matching the target value.

3.3.2. *Target Aware AutoInflaters - [high population, low-dimension data]*

Autoencoders are commonly used on high-dimensional data to reduce the number of features and extract valuable information. This is why they are often applied to image datasets. However, in real-life tabular regression data, the number of features is typically not too large, and compressing the data may result in loss of information. To address this issue, we propose an expansion-decoder network specifically designed for datasets with a lower number of features. Fig. 17 illustrates this design, where the initial part of the network **inflates** the feature information into a **basin** and then **deflates** it back to both the features and the target value. This approach allows us to preserve the information in the data without distortion.

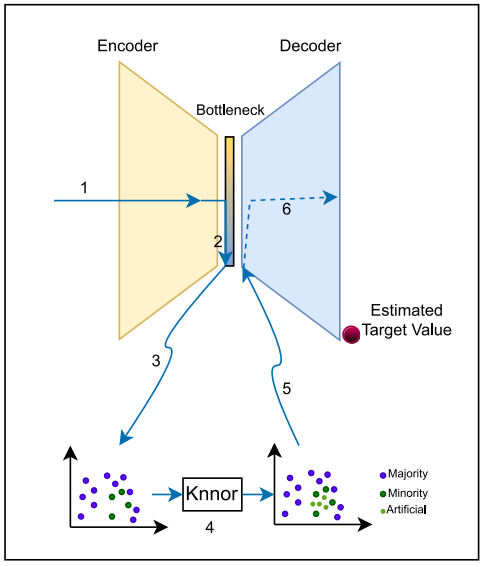


Fig. 18. Role of Target Aware Auto Encoder in generating new data samples as well as estimating target value for the novel data points. Same can be applied to target aware auto-inflater.

The ultimate objective of both the target-aware autoencoder and deflater is identical. They aim to learn dataset features in a manner that the bottleneck representation can be used to generate new data points while the decoder can be used to obtain target values for any data point. This process is illustrated in Fig. 18, depicting the sequential steps from 1 to 6. Initially, a target-aware autoencoder/inflater is trained. Step 1 represents passing the data as input to the system. Bottleneck features are extracted at step 2 and passed to the KNNOR Regression algorithm in steps 3 and 4 to generate new data points. The decoder is leveraged

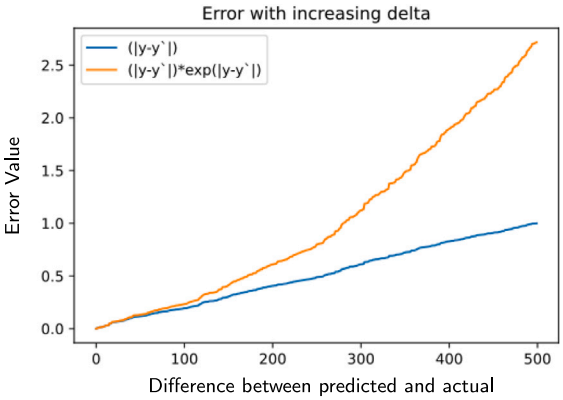


Fig. 19. Progression of error values with increasing distance between actual and predicted.

at steps 5 and 6 to estimate the target value for the artificial data point created.

3.3.3. Exponential loss function [Applicable to Target Aware Auto Encoders/Inflaters]

To assess the accuracy of both the AutoInflaters and Target-Aware AutoEncoders, common regression loss functions such as Mean Absolute Error (MAE) or Root Mean Square Error (RMSE) can be employed. The overall loss is reiterated in Eq. (8) as follows:

$$total\_loss = \mathcal{L}(\hat{X}, X) + W * \hat{\mathcal{L}}(\hat{y}, y) \tag{8}$$

where $total\_loss$ is the accumulated loss over the features and the target values. The $\hat{\mathcal{L}}$ function can be same as the function $\mathcal{L}$ applied on the feature set, however we propose the two functions to be separate. As the count of features outnumbers the target variable, the network is coerced to prioritize the target variable by:

- Using a penalty function $\hat{\mathcal{L}}$ on the target values such that the difference between the predicted and actual target value is magnified;
- Adding W to the above loss value to balance for the disparity between cardinality of feature sets (>> 1) and cardinality of target values (usually one);
- By introducing a weight value and employing a distinct loss function $\hat{\mathcal{L}}$, we aim to incorporate a penalizing function that exhibits accelerated growth as the deviation from the original value increases. This function is depicted in Eq. (9) as follows.

$$\hat{\mathcal{L}}(\hat{y}, y) = |\hat{y} - y| * e^{|\hat{y}-y|} \tag{9}$$

The efficacy of the function is described in Fig. 19. The orange line depicts 1000 randomly generated delta values or the absolute difference between the predicted and actual, while the blue line plots the penalty value $\mathcal{L}$. This additional loss ($\hat{\mathcal{L}}$) calculation along with the weight factor (W) helps in balancing the influence of the input features over the target variable.

3.4. Multi-level Auto Encoder - [Image data]

In the case of image datasets, the approach was different as training a target-aware Auto Encoder directly on the images was not yielding good results. The reason was that the image features' loss function overwhelmed the target variables' loss function. The accuracy of the entire model was high despite the accuracy of target prediction being relatively low. In order to cope with this discrepancy, a multi-level Autencoder scheme is proposed. The first-level auto-encoder is a simple Convolution Autoencoder that extracts features from the images and converts them into vectors at the bottleneck. At the second level lies a target aware, fully connected Auto Encoder that uses the bottleneck of the previous level Autoencoder as input and trains a target-aware Neural Network. Fig. 20 illustrates the process. The external Convolution Auto Encoder is responsible for extracting the features from the image dataset at the bottleneck (marked as BottleNeck1 in Fig. 20). These features and the target values (provided externally) are used to train the internal target-aware Auto-Encoder. The bottleneck of the internal Auto-Encoder (marked as BottleNeck2) is used to reduce the feature size of the dataset further, and KNNOR is applied to these extracted features to generate new data points. This approach proves to be more efficient than training a single target-aware autoencoder on the images directly, as seen in Table 7.

3.4.1. Approach summary

The approach for estimating the target value of artificial data points needs to adapt as the shape of the data changes. In this regard, two key characteristics are taken into account: the number of features in the data and the population of the minority dataset. These factors play a significant role in determining the appropriate method for estimating the target value in the context of generating artificial data points.

- *Large Dataset with a high number of features*. In this case, we employ a Target Aware autoencoder (depicted in Fig. 16) to extract the features and reduce their dimensionality. Subsequently, the KNNOR algorithm is applied to this feature set in order to upsample the minority dataset. Finally, we utilize the pre-trained Target Aware autoencoder to predict the target values for the generated artificial data points. The process is illustrated in Fig. 21. This approach is commonly utilized for numerous image datasets.
- *Large Dataset with low number of features*. In this case, we employ a Target aware Inflate-Deflate architecture to expand the dataset and enhance its representation. The features are extracted from the basin and utilized to generate artificial data points using the KNNOR algorithm. Subsequently, these data points are fed into the Deflater part of the network to generate the corresponding target data points. Fig. 22 illustrates this process. This approach is particularly useful for large tabular datasets.
- *Small Dataset with a high or low number of features*. In case where the dataset is small, we employ the KNNOR algorithm to create artificial data points regardless of the number of features. Subsequently, we utilize the KNNOR-Reg method to predict the potential target values for these artificial data points. The steps involved in this process are illustrated in Fig. 23.

The result of augmentation process for different datasets has been shown in Figs. 24 and 25. Figures a and b of each figure show a scatter plot of the data after doing a Principal Component Analysis (PCA) for representation in 2-dimensions.

Fig. 24(a and b) shows the augmentation efforts on the laser dataset using the KNNOR-Regression method (Section 3.2). Fig. 24(c and d) illustrate a variant of the KNNOR-DeepRegression method where KNNOR is used to oversample the data and then an AutoInflater is used to estimate the target values for the synthetic data. In Figs. 24a and 24c, the augmented points are the same, however since the method to obtain the target value is different in each case, Figs. 24b and 24d capture a different distribution of the augmented target values. Fig. 25 shows the same comparison for the ele-2 dataset using the same 2 methods. Here also, Fig. 25b shows the target values obtained using the target values of the k-nearest neighbors. In case of Fig. 25d, a Target Aware AutoInflater was trained on the training data. Consequently the new points generated using KNNOR were passed into the AutoInflater to obtain their target values. The difference is apparent in the histogram of distributions as shown in Figs. 25b and 25d, where, although the shapes of the histograms are similar, the frequency of the different ranges of values is different.

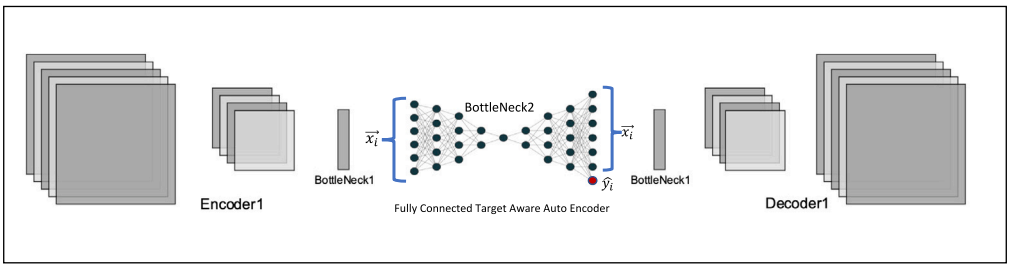


Fig. 20. Multi-level Auto Encoders for Images. First, CNN Auto Encoders to reduce the dimension of images. Second, Fully Connected Target Aware Auto Encoders to learn the target values of minority data set.

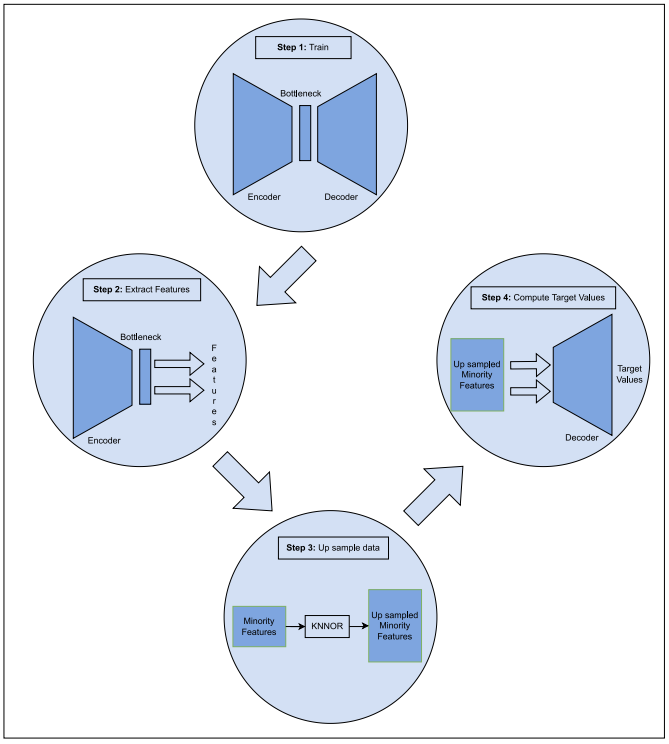


Fig. 21. Target generation steps for data with high population and high dimension.

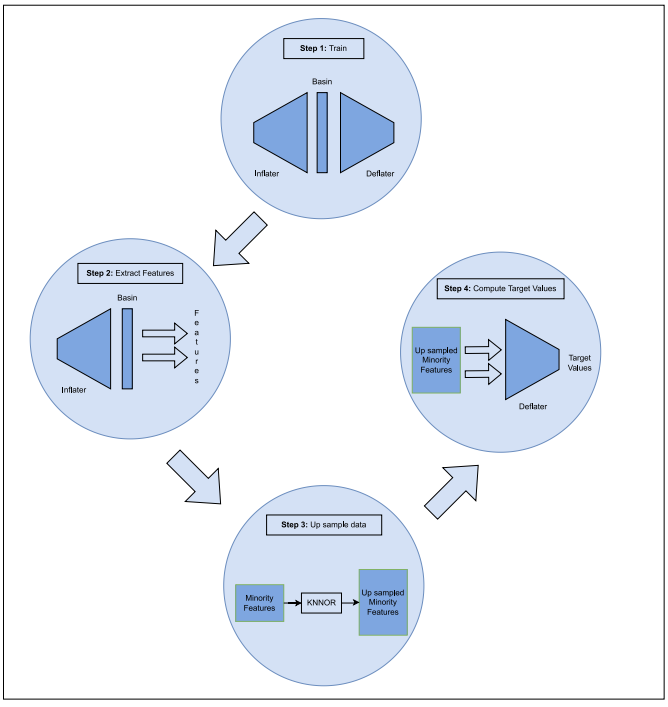


Fig. 22. Target generation steps for data with high population and low dimension.

3.5. Summary

Fig. 26 provides an overview of the various approaches mentioned in the previous subsections, allowing users to easily assess and select the most suitable method based on the data structure being considered. The figure showcases the approach used for image data, while for tabular data, specific recommendations are provided for scenarios involving high or low population and high or low dimensions. This comprehensive visualization aids in navigating the available techniques and making informed choices based on the characteristics of the data at hand.

4. Results and discussion

In this section, we detail our experimental design, where we evaluate the efficacy of our methods on established imbalanced regression datasets and present the results and discussions. We have delineated four distinct sets of experiments. The first experiment involves a performance comparison between our oversampling technique and SMOTER on tabular data, while the second experiment replicates this performance evaluation using image data. The third experiment assesses whether our augmentation method genuinely enhances the regression capabilities of three state-of-the-art regressors. Lastly, the fourth experiment gauges the efficiency of our oversampling algorithm in improving the predictive prowess of 18 regressors.

4.1. Experiment design

This section focuses on evaluating the effectiveness of oversampling techniques and target value predictors in the subsequent regression step. The experimental procedure is designed to compare the proposed methods in this paper against existing techniques, with a primary emphasis on SMOTER (Torgo, Ribeiro, Pfahringer, & Branco, 2013) for both tabular and high-dimensional (image) datasets. 1 and 2 provide details on the datasets used, sourced from the data repository at https://paobranco.github.io/DataSets-IR/ (Branco et al., 2019), and the Keel repository (Derrac, Garcia, Sanchez, & Herrera, 2015). 33% of the data in each file was preserved for test while the rest was used for training and oversampling. For these tabular datasets, we applied the approaches illustrated in Figs. 22 and 23. To assess the performance on high-dimensional datasets, we utilized the Image-Age dataset obtained from the AgeDB (Moschoglou et al., 2017) and IMDB-WIKI (Rothe, Timofte, & Van Gool, 2018) repositories, employing the approach illustrated in Fig. 21. In the case of AgeDB, images with an age label exceeding 80 were considered rare, while for IMDB-WIKI, the

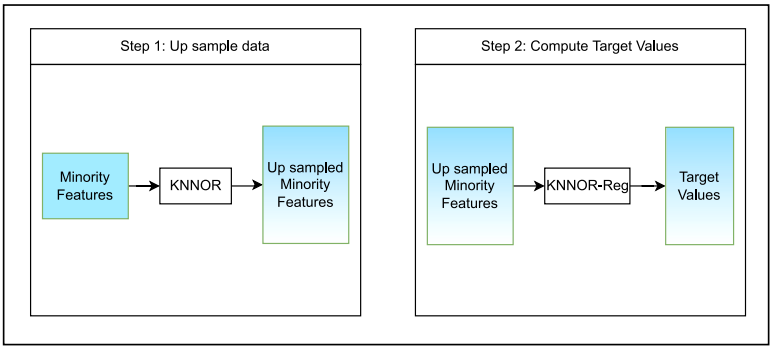


Fig. 23. Target generation steps for data with low population.

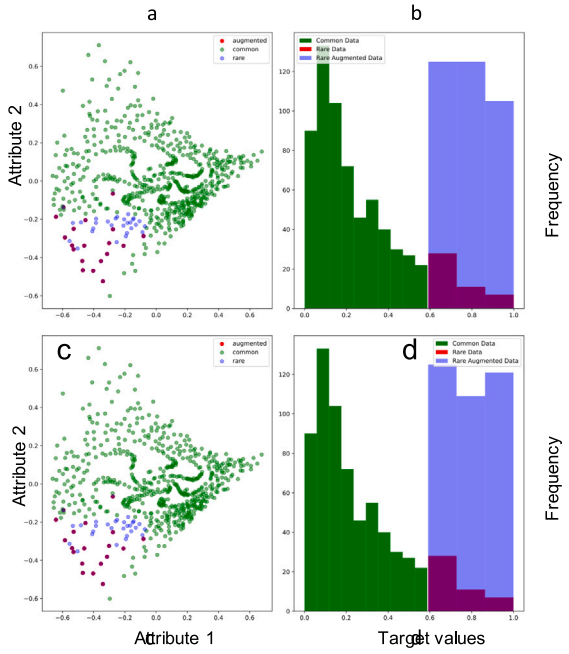


Fig. 24. Augmentation of data and target values on the laser dataset, using KNNOR-Regression and KNNOR-DeepRegression. Figures a and c show the scatter plot of data considered to be common and rare and also show the datapoints after augmentation, Figures b and d show the frequency of the labels of the common, rare and augmented data points.

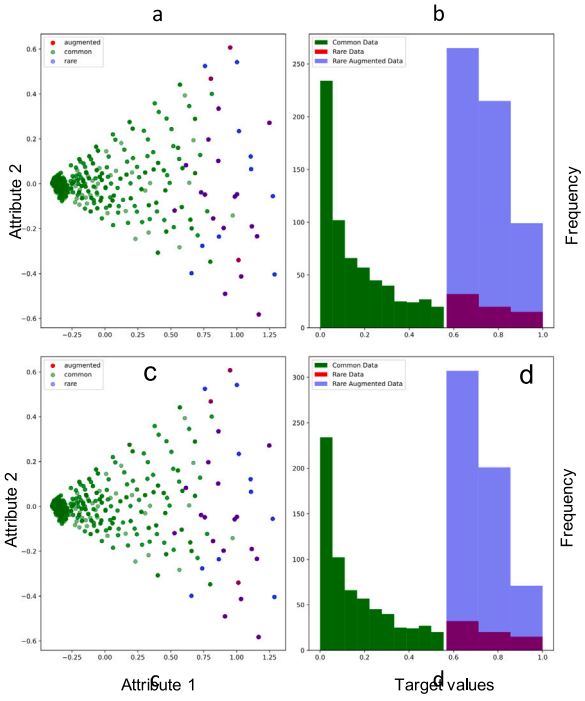


Fig. 25. Augmentation of data and target values on the ele-2 dataset, using KNNOR-Regression and KNNOR-DeepRegression. Figures a and c display scatter plots representing data points categorized as common and rare, including the augmented data points. Meanwhile, Figures b and d illustrate the label distribution for common, rare, and augmented data points.

age threshold was set at 75. By conducting these experiments, we aim to provide a comprehensive comparison and evaluation of our proposed methods alongside existing techniques.

4.1.1. Evaluation process and metrics

The dataset was initially divided into training and test sets, with the test dataset being kept separate throughout the augmentation and training process. The three methods employed are outlined as follows:

1. The training data was split into two classes, rare and common, depending on the relevance threshold. Considering it to be a classification dataset (rare and common), it was passed through the KNNOR approach (Islam et al., 2022b) approach to create artificial data points of the rare category. The target labels of points used in the creation of each artificial data point were then aggregated to calculate the label of the new point created. The augmented data points with labels were added back to the training set, and regressors were trained to check the performance on the test dataset;
2. KNNOR-DeepReg process: Similar to the KNNOR-Reg process, the training data was split into rare and common classes. The rare data was used to train target-aware autoencoders (depicted in Figs. 16 and 17). The KNNOR method was then applied separately to create artificial data points for the rare class. These generated artificial points were passed through the trained autoencoders to obtain the target labels for the new data points. The augmented data points, along with their labels, were incorporated back into the training set, and regressors were trained to assess performance on the test dataset;
3. For image datasets, the process began by dividing the training data into rare and common classes. A generic autoencoder was trained on the images to learn and extract their features, thereby reducing the dimensionality of the data. The entire training image set was then fed through the autoencoder to extract features for both rare and common data. A target-aware autoencoder was trained on the extracted features of the rare class to learn the target values specific to those features. KNNOR was

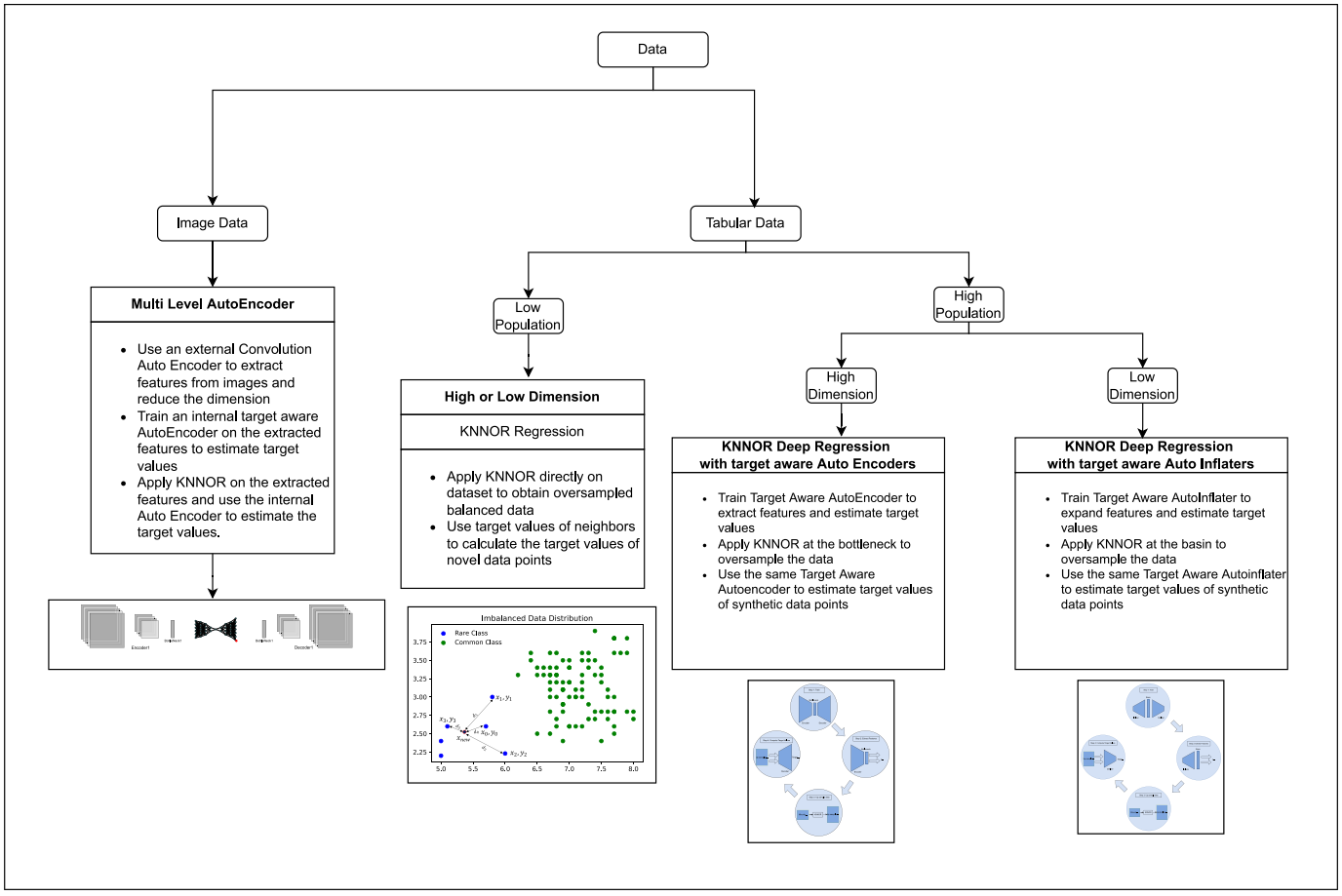


Fig. 26. Recommendation of techniques for varying structure of data. Regarding Tabular data, we offer different methods depending on the population and number of features. We propose KNNOR-Regression (Section 3.2) for low population data. For data with a high population, we advocate KNNOR Deep Regression which has two flavors. For high-population data with a high number of features, we use Target Aware AutoEncoders (Section 3.3.1), and for high-population data with a low number of features, we use Auto Inflaters (Section 3.3.2). Finally, for Image datasets, we use a combination of Convolution and Fully Connected AutoEncoders called Multi-Level AutoEncoders (Section 3.4).

Table 1
Numerical datasets used in comparison.

Data set	Instances	Features	Relevance threshold	Rare	Rare (percentage)	Type of extreme
ANACALT	4052	7	0.8	835	0.21	lower
bank8FM	4499	8	0.8	285	0.06	upper
baseball	337	16	0.5	50	0.15	upper
boston	506	13	0.8	113	0.22	upper
compactiv	8192	21	0.8	713	0.09	lower
concrete	1030	8	0.8	52	0.05	upper
cpuSm	8192	12	0.8	713	0.09	lower
ele-1	495	2	0.8	43	0.09	upper
ele-2	1056	4	0.8	110	0.1	upper
forestFires	517	12	0.8	78	0.15	upper
friedman	1200	5	0.5	48	0.04	upper
laser	993	4	0.8	75	0.08	upper
machineCPU	209	6	0.8	31	0.15	upper
mortgage	1049	15	0.8	106	0.1	upper
quake	2178	3	0.8	118	0.05	upper
stock	950	9	0.5	63	0.07	upper
treasury	1049	15	0.8	109	0.1	upper
wankara	321	9	0.5	31	0.1	lower

Table 2
High dimension (image) datasets used in comparison.

Data set	Instances	Features	Rare	Rare (percentage)	Type of extreme
AgeDB	16 488	64 × 64	494	0.03	upper
IMDB-WIKI	213 553	64 × 64	2721	0.012	upper

subsequently applied to the training set of extracted features, creating additional artificial data points belonging to the rare class. The augmented features were then passed through the target-aware autoencoder to generate the corresponding target

Table 3
Hyperparameters of regressors.

Regressor	Hyperparameters
Support Vector regression	kernel = 'rbf', degree = 3, gamma = 'scale', coef0 = 0.0, tol = 0.001, C = 1.0
Random Forest regression	n_estimators = 100, criterion = 'squared_error', max_depth = None, min_samples_split = 2, min_samples_leaf = 1
Gradient Boosting regression	loss = 'squared_error', learning_rate = 0.1, n_estimators = 100, subsample = 1.0, criterion = 'friedman_mse', min_samples_split = 2, min_samples_leaf = 1
Fully Connected Network	hidden_layer_sizes = (100,), activation = 'relu', *, solver = 'adam', alpha = 0.0001, batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001

Table 4
Demonstration of relative error. The two rows represent two different distributions. For the first distribution, maximum of test target values is 0.1 while that for the second is 100. Error percentage is Absolute Difference divided by the maximum of true target values for each distribution.

True value	Predicted value	Absolute difference	Error percentage
0.1	1.1	1	0.909
100	101	1	0.0099

values. The augmented data points, along with their generated labels, were integrated back into the training set of extracted features. Regressors were trained and applied to the test data to evaluate accuracy.

By employing these three methods, the performance and accuracy of each approach were assessed using the test dataset. It can be noted that the third method is an extension of the second method as it contains an additional step of feature extraction from image datasets. Also the processes of transformation on the training data have been applied to the test data to enable accuracy estimation. The machine learning algorithms used at the end include Linear Regression (Barupal & Fiehn, 2019), Support Vector regression (SVR) (Vapnik & Vapnik, 1998), Random Forest regression (RF) (Segal, 2004), Gradient Boosting regression (GBR) (Natekin & Knoll, 2013) and a Fully Connected Network (FCN) (Kohler & Langer, 2021). The fully connected network consists of the input layer, followed by 5 hidden layers and the output layer. The activation function is Rectified Linear Units (ReLU) (Agarap, 2018) in each case. Table 3 defines the different hyper parameters for each regressor.

The RMSE has been used in calculating the error percentage overall. Although RMSE is a ubiquitous error metric for regression, it does not lend itself easily to comparison for results on normalized data. The data needs to be denormalized before calculating RMSE. We propose a different relative error calculation method on the normalized output itself in order to obtain more intuitive and easily comparable results. The relative max error (RMaxE) is defined in Eq. (10) as follows:

$$RMaxE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{\hat{y}_i - y_i}{y_{max}}\right)^2} \tag{10}$$

where $\hat{y}_i$ is the predicted value and y_i is the actual target value. The value y_{max} is the maximum target value in the test set, and n is the number of samples in the same. The utility of the error metric is illustrated in Table 4, which shows the difference in error percentage even when the absolute differences are the same. A true value of 0.1, represented as 1.1, is inferior to predicting 101 against the true value of 100. In order to allow the error to be expounded, we divide the absolute difference by the maximum true value in the test set. This helps magnify the error at lower values compared to the same error at higher values.

4.1.2. Augmentation framework

For each train–test split of every dataset, we conduct evaluations with varying levels of augmentation employing different regression techniques. The initial evaluation is conducted without any augmentation, and subsequent assessments involve the application of both state-of-the-art and our proposed augmentation methods.

- **State-of-art:** In this context, we utilize the SMOTE for Regression (SMOTER) technique, which is introduced by Torgo (Torgo et al., 2013), as our chosen augmentation method. The parameters configured for SMOTER are as follows:
  - Maximum distance: The maximum distance from existing minority points, with values ranging from 0.001 to 0.01 to 0.1. This determines the placement of new points relative to the existing minority points;
  - Addition proportion: A list of values ranging from 0.1 to 0.5 to 0.8, indicating the proportion of the population to be added to the minority dataset. For example, an addition proportion of 0.1 means that, after adding the minority points, the minority population would be 60% of the majority population. It is worth noting that, unlike (Camacho et al., 2022), we did not aim to match the minority population with the majority population exactly, to save computational effort and showcase comparative improvement over the original data (Haixiang et al., 2017).
- **Proposed Approach**: In this case, we employ the KNNOR approach (Islam et al., 2022b) as the augmentation technique. The parameters used in KNNOR are as follows:
  - Number of neighbors: The number of neighboring points used to generate a new data point, with values ranging from 2 to 5 to 10;
  - Usable minority proportion: The proportion of the minority population used in generating the artificial data point, with values ranging from 0.2 to 0.6 to 0.9. For example, if the minority population is 100 and the usable minority proportion is 0.6, only 60 minority data points are utilized in producing the artificial data points. The selection of these 60 data points is based on a criticality estimate explained in Islam and Belhaouari (2021).

With the range of parameters mentioned, a total of 243 experiments are conducted on each file, resulting in a cumulative total of 53,217 experiments for the 18 numerical datasets.

4.2. Results

Tables 5, 6, and 7 provide the results of the conducted comparisons. Table 5 presents the mean of the best performance achieved by various regressors on different datasets, with the max-relative-score serving as the error metric. In Table 6, the mean Root Mean Squared Error (RMSE) score on different datasets is presented. Table 7 focuses specifically on the RMSE score for image datasets. Each column in these tables represents a different technique employed in the experiment, while the rows correspond to the various regression algorithms utilized. It is important to mention that in Table 6, the AutoInflaters have been trained with either the exponential or RMSE loss, corresponding to the columns in Table 5. In Tables 5 and 6, the columns are structured as follows.

- Column 1: Different Regressors used;

Table 5
Relative-max-error scores for different datasets.

Regression algorithm	No oversampling	SMOTE regression	Type I	Type II	Type III	Type IV	Type V	Type VI	Type VII
RFR	1.1749	1.1397	**1.1212**	1.1263	1.135	1.2892	1.2589	1.2844	1.3244
GBR	1.1479	1.1184	**1.0924**	1.1054	1.1055	1.3297	1.2951	1.3291	1.3526
SVR	1.7765	1.7095	1.662	1.6663	1.6647	1.629	1.6316	**1.6069**	1.6902
LR	2.0867	2.1257	2.1056	2.1047	2.1114	1.6612	**0.4964**	1.6502	1.5737
FCN	0.3951	0.4886	0.4866	**0.3883**	0.3915	0.5244	0.5665	0.5187	0.5676

Table 6
RMSE scores for different datasets.

Regression algorithm	No oversampling	SMOTE regression	Type I	Type II	Type III	Type IV	Type V	Type VI	Type VII
RFR	0.0085	0.0075	**0.0071**	0.0074	0.0073	0.0097	0.0095	0.0097	0.0091
GBR	0.0078	0.0074	0.0065	0.0065	**0.0064**	0.0098	0.0087	0.0097	0.0092
SVR	0.013	0.0126	0.0119	0.0118	0.0117	0.0117	0.0117	**0.0115**	0.0116
LR	0.0122	0.0121	0.0118	0.0119	0.0118	0.011	**0.0092**	0.0107	0.0095
FCN	0.0066	0.006	**0.0058**	0.0064	0.0064	0.0121	0.0126	0.0121	0.0125

- Column 2: Regression error when no augmentation is applied;
- Column 3: Regression error when state of the art SMOTE-Regression is applied;
- Column 4 **(Type I/KNNOR-Regression)**: Regression error after applying KNNOR to create synthetic datapoints and then applying KNNOR-Regression (Ref 3.2) to determine the target values of the newly created points;
- Column 5 **(Type II)**: Initially, KNNOR is used to generate artificial data points. Subsequently, the Target-Aware AutoInflaters (Ref 3.3.2) trained on the training data are employed to estimate the target values for these newly created data points. The loss function used in these AutoInflaters is RMSE;
- Column 6 **(Type III)**: This column represents the same process as the previous one, with the difference being that the loss function used in these AutoInflaters is the exponential loss function specified in 3.3.3;
- Column 7 **(Type IV)**: In this scenario, KNNOR is applied to the expanded features of the training data obtained using the first part of the AutoInflaters. The target values are determined using the latter (Deflater) part of the same AutoInflaters. The loss function used in these AutoInflaters is RMSE;
- Column 8 **(Type V)**: This column follows a similar process to the previous one, but the loss function used in the AutoInflaters is the exponential loss function specified in 3.3.3;
- Column 9 **(Type VI)**: In this case, the AutoInflaters are employed to extract the features. Subsequently, KNNOR-Regression is applied to the inflated features to create the artificial dataset as well as the target data points. The loss function used in the AutoInflaters is RMSE;
- Column 10 **(Type VII)**: This column is similar to the previous one, with the difference being that the loss function used in the AutoInflaters is the exponential loss function specified in 3.3.3.

The distribution of ranks achieved by non-augmented methods and augmentation methods, including SMOTER, KNNOR-Regression, and KNNOR-DeepReg, is visualized in Figs. 27 and 28. Fig. 27 displays the ranks obtained when the regression metric was relative max error, while Fig. 28 presents the ranks of the various regressors when RMSE was the evaluation metric. The variants of KNNOR-Regression have ranked better a higher number of times than the SMOTE-Regressor.

In both Figs. 27 and 28, the lines represent different algorithms, and they are organized as follows:

- **No Augmentation**. Represents the frequency of ranks for regressors trained on non-augmented data;
- **SMOTE-Regression**. Represents the frequency of ranks for regressors trained on data augmented by the SMOTER method;

Table 7
RMSE score for the image datasets using a fully connected regressor.

Dataset	Error metric	No augmentation	SMOTER	KNNOR DeepReg
IMDB-WIKI	RMSE	0.709	0.153	**0.137**
	Relative-Max	3.469	1.657	**1.567**
AgeDB	RMSE	2.33	0.296	**0.294**
	Relative-Max	3.328	1.183	**1.178**

- **KNNOR-Regression**. Represents the frequency of ranks for regressors trained on data augmented by KNNOR and target values calculated by the KNNOR-Reg method 3.2;
- **KNNOR-DR-I**. Represents the frequency of ranks for regressors trained on data augmented by KNNOR and target values calculated by using the target aware AutoInflaters 3.3;
- **KNNOR-DR-II**. In this case KNNOR was applied on features extracted by the target aware AutoInflaters. The same AutoInflaters were used to calculate the target values;
- **KNNOR-DR-III**. In this case KNNOR was applied on features extracted by the target aware AutoInflaters. KNNOR-Reg was used on the artificial data point to calculate the target values.

KNNOR-Regression achieves the best rank among all, followed by different variations of KNNOR-DeepRegression (KNNOR-DR-X).

Table 7 shows the RMSE score for the two Image datasets used in the experiment. As the number of datapoints and features are considerably high, we have only tested them on fully connected deep regressors. Data imputation increases the accuracy of the models manifolds, with KNNOR-DeepReg, the accuracy is enhanced further.

The KNNOR-Regression method demonstrates superior performance across the numerical datasets, followed by the KNNOR-DeepReg methods. We believe that as the dataset size increases, the KNNOR-DeepReg method, which utilizes Target Aware AutoInflaters, has the potential to outperform the KNNOR-Reg process.

The overall performance, as indicated by the RMSE metric, is consistent with the results obtained using the Relative-max-error metric, confirming its validity. Furthermore, the range of scores in Table 4 (using the relative-max-error metric) is more pronounced and easier to compare than in Table 6, where the error values show differences in the third or fourth decimal place.

4.3. Experimenting with strong and weak regressors

In a recent scientific paper titled "To SMOTE, or not to SMOTE?" (Elor & Averbuch-Elor, 2022), the benefits of balancing techniques are explored, particularly in relation to advanced classifiers. The study

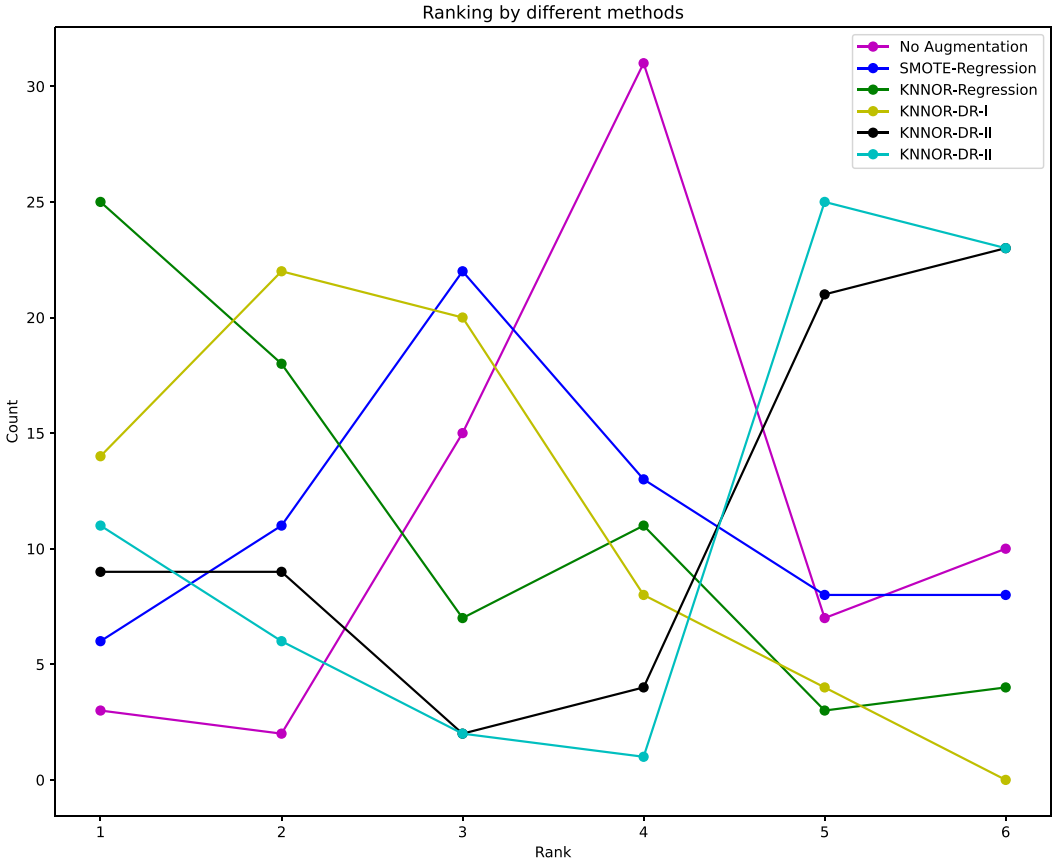


Fig. 27. Ranks of different methods on all datasets when the error metric was max-relative-error. Lines represent the different methods explained in Section 4.2. KNNOR-Regression denoted by the green line is the best performing method having ranked the maximum number of times.

involves conducting extensive experiments using three state-of-the-art classifiers, comparing them to weaker learners used in previous studies. The results reveal that balancing techniques significantly improved the prediction accuracy of weak classifiers, including multi-layer perceptron, support vector machines, decision tree, and Adaptive Boosting (AdaBoost). However, no noticeable impact is observed on the performance of more robust classifiers like eXtreme Gradient Boosting (XGBoost) and categorical boosting (Catboost).

To assess the efficacy of the methodology introduced in our paper, we extend our analysis to include the regression variants of the advanced predictors mentioned in the aforementioned publication. We compare their performance on both the original and augmented datasets, observing notable performance improvements following augmentation. In our assessment, we utilize a comprehensive set of regression metrics for evaluation including the following:

- **MAE** measures the average absolute difference between predicted and actual values;
- **Mean Squared Error (MSE)**. MSE calculates the average squared difference between predicted and actual values;
- **R-squared (R2)**. R2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates a perfect fit;
- **Explained Variance Score**. This metric quantifies the proportion of the variance in the target variable that is explained by the model. A score of 1 indicates a perfect fit;
- **Median Absolute Error (MedAE)**. Quantifies the median magnitude of errors between predicted and actual values, offering robustness to outliers.

In Table 8, the "Wins" column shows how many times the state-of-the-art (SOTA) regressor on augmented data outperformed the SOTA regressor on the original data. This count is significantly higher implying that the augmentation of imbalanced data using our technique enhances the performance of strong regressors. For those interested in exploring the underlying code and details of this simulation, we have made the code repository available on GitHub at the following Github link.

Table 8
Comparing strong regressors on Augmented vs. Original data. Column "Wins" indicates how many times the performance of regressor on augmented data was better than regressor on original imbalanced data.

Regressor	Error metric	Wins	Losses
CatBoost	Mean Absolute Error	9	7
	Mean Squared Error	11	5
	R-squared	11	5
	Explained Variance Score	11	5
	Median Absolute Error	10	6
Total		52	28
XGBoost	Mean Absolute Error	12	4
	Mean Squared Error	10	6
	R-squared	10	6
	Explained Variance Score	11	5
	Median Absolute Error	13	3
Total		56	24
LightGBM	Mean Absolute Error	**13**	3
	Mean Squared Error	**14**	2
	R-squared	**14**	2
	Explained Variance Score	**14**	2
	Median Absolute Error	**12**	4
Total		**67**	13

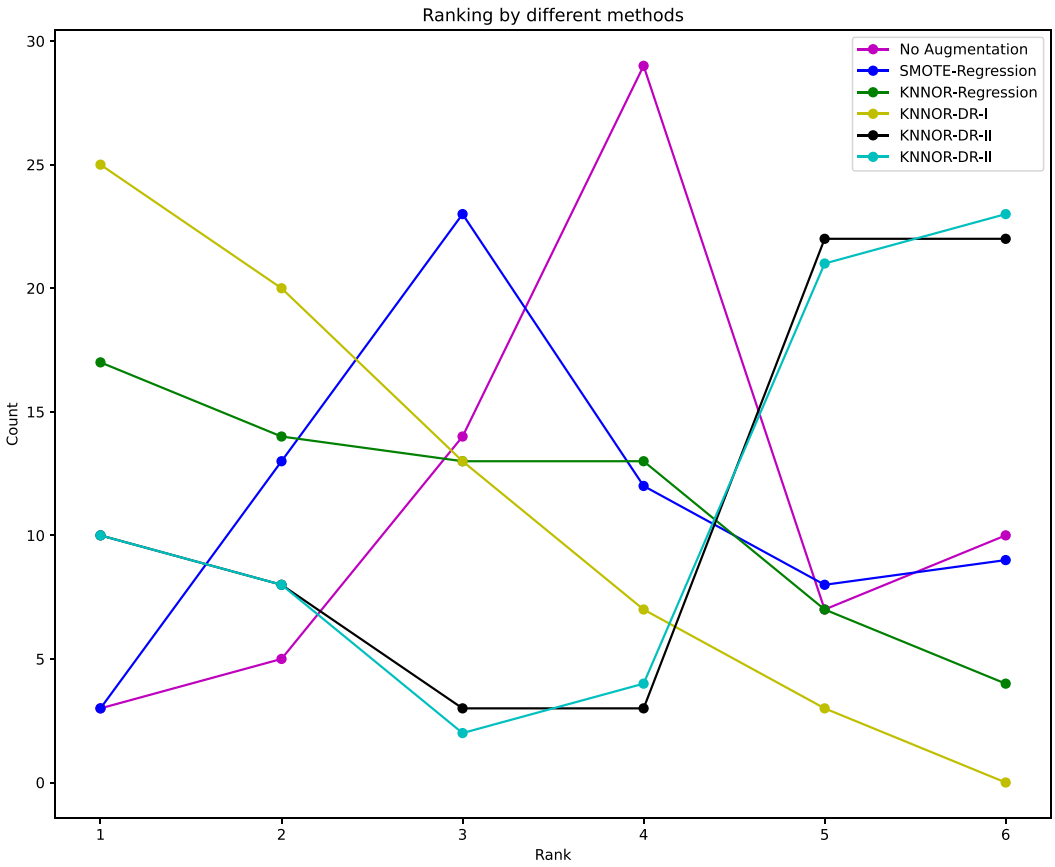


Fig. 28. Ranks of different methods on all datasets when the error metric was RMSE. Lines represent the different methods explained in Section 4.2. KNNOR-DR-I is the best performing method followed by the KNNOR-Regression method as they rank 1st maximum number of times.

Table 9
Models used for extensive testing. All of them have been used by leveraging the Pycaret library in python which gives standard implementation of all the models (Ali, 2020).

Models	Models	Models
AdaBoost Regressor	Extreme Gradient Boosting	Least Angle Regression
Bayesian Ridge	Gradient Boosting Regressor	Light Gradient Boosting Machine
Decision Tree Regressor	Huber Regressor	Linear Regression
Dummy Regressor	K Neighbors Regressor	Orthogonal Matching Pursuit
Elastic Net	Lasso Least Angle Regression	Passive Aggressive Regressor
Extra Trees Regressor	Lasso Regression	Random Forest Regressor

4.4. Testing with additional models

In order to cement our claims, we have added a host of models to train on augmented as well as non-augmented data and then compare their performance on the test data. The datasets have been already mentioned in Table 1 and the models used are mentioned in Table 9. Default hyper-parameters were used for each model as mentioned in Ali (2020).

We conducted a comprehensive evaluation by applying each model to both augmented and non-augmented datasets. To assess their performance rigorously, we employed a set of six essential metrics: MAE, MSE, RMSE, R-squared (R2), Root Mean Squared Logarithmic Error (RMSLE), and Mean Absolute Percentage Error (MAPE). These metrics have been defined in Section 4.3 and provide a well-rounded perspective on how well the models make predictions.

Our analysis involved calculating how many times the models trained on augmented data outperformed those trained on non-augmented data across all six metrics. This cumulative count provides a consolidated measure of the augmentation's impact on model performance. By considering multiple metrics and aggregating the results, we gain a holistic understanding of the benefits of using augmented data, offering a robust assessment of its efficacy in enhancing predictive models. A total of 1368 experiments were made, for 18 models on 18 datasets. Augmentation was performed for 3, 4 and 5 neighbors and the best result was selected.

The results in Table 10 provide a comparative overview of 18 regression models evaluated on both augmented and original imbalanced data. Following is a detailed summary of the findings:

1. Passive Aggressive Regressor, Linear Regression, and Light Gradient Boosting Machine are the top-performing models, with wins in approximately 83.33% of cases.
2. Ridge Regression and Bayesian Ridge follow closely, with wins in about 82.29% of cases.
3. Decision Tree Regressor, Random Forest Regressor, and AdaBoost Regressor also demonstrate strong performance, with wins in 79.17% to 75.15% of cases.
4. Several models, including Extreme Gradient Boosting, Least Angle Regression, and Lasso Regression, exhibit wins in approximately 75% of cases.
5. Models like Extra Trees Regressor, Huber Regressor, and Elastic Net have wins in the range of 68.75% to 66.67% of cases.

Table 10
Comparing 18 regression models on Augmented vs. Original Data. Column "Wins" indicates how many times the performance of regressor on augmented data was better than regressor on original imbalanced data.

Regressor	Wins	Losses	Percentage Wins (%)
Passive Aggressive Regressor	**480**	96	83.33
Linear Regression	**480**	96	83.33
Light Gradient Boosting Machine	**480**	96	83.33
Ridge Regression	474	102	82.29
Bayesian Ridge	474	102	82.29
Decision Tree Regressor	456	120	79.17
Random Forest Regressor	444	132	77.08
AdaBoost Regressor	517	171	75.15
Extreme Gradient Boosting	432	144	75
Least Angle Regression	432	144	75
Lasso Regression	414	162	71.88
Extra Trees Regressor	402	174	69.79
Huber Regressor	396	180	68.75
Elastic Net	384	192	66.67
Gradient Boosting Regressor	366	210	63.54
Orthogonal Matching Pursuit	324	252	56.25
K Neighbors Regressor	318	258	55.21
Lasso Least Angle Regression	228	348	39.58
Total	**7501**	2979	71.57

6. Gradient Boosting Regressor, Orthogonal Matching Pursuit, and K Neighbors Regressor have relatively lower win percentages, ranging from 63.54% to 55.21%.
7. Lasso Least Angle Regression has the lowest win percentage at 39.58%.

The overall trend indicates that several regression models benefit from using augmented data, resulting in a performance improvement. Passive Aggressive Regressor, Linear Regression, and Light Gradient Boosting Machine consistently outperform others, showcasing their robustness in handling imbalanced data when augmented. However, it is important to note that the degree of improvement varies among the models, and some models may not benefit significantly from augmentation. The aggregate win percentage across all models is 71.57%, underscoring the overall efficacy of data augmentation in enhancing regression model performance. The code base for comparison is available at the following Github link.

5. Conclusion and future work

This paper introduces innovative techniques for augmenting regression datasets, specifically addressing the challenge of imbalanced data. We extend the widely used K Nearest Neighbor OverSampling (KNNOR) approach, commonly employed in imbalanced classification datasets, to work effectively in regression datasets. Unlike classification data, regression datasets require estimating continuous target values for newly created data points. To tackle this, we enhance the KNNOR algorithm by keeping track of the neighboring data points used in generating new data points and accumulating their corresponding target values. The target value of the newly created point is then estimated based on the distances to these neighboring points.

Furthermore, we propose the use of Target Aware Autoencoders and Target Aware AutoInflaters as an alternative approach for estimating target values in regression datasets. This involves incorporating a weighted component of target estimation into the loss equation while training the autoencoders. By learning the features of the rare dataset and fine-tuning the weights, these models can predict continuous values of the target variable.

In addition to numerical datasets, we also introduce a two-tier autoencoder scheme tailored for imbalanced image datasets. This scheme enables the extraction of features and learning of target values at subsequent levels of autoencoders, allowing for effective augmentation of imbalanced image data.

Our experimental results clearly demonstrate the superior performance of the proposed approaches in comparison to previous state-of-the-art methods. In an effort to facilitate the adoption of these techniques, we provide the code as a user-friendly tool, accessible even to non-specialist users. These methods hold significant potential for improving critical areas of regression where imbalanced data presents a significant challenge.

To make our code readily accessible, we host it on GitHub and created a Python package. This code primarily focuses on the core features of our concept, particularly in managing imbalanced numeric datasets by applying distribution-aware oversampling. The code base can be found at Github and the python package can be installed from pypi. A jupyternotebook is also available to the reader as an end-to-end implementation of the algorithm.

5.1. Future research directions

Building on the findings of this study, the following research endeavors can be pursued:

1. Expansion to Other Data Types: While this paper focuses on numerical and image datasets, future work could explore the application of these augmentation techniques to other data types, such as time-series or audio data, where imbalance also poses significant challenges.
2. Integration with Deep Learning Models: Further research could investigate the integration of the proposed augmentation methods directly within deep learning training pipelines, potentially enhancing model performance by providing more balanced and informative training data.
3. Automated Feature and Target Value Adjustment: Developing algorithms that automatically adjust feature representations and target values based on the specific characteristics of the dataset could lead to more generalized and effective augmentation strategies.
4. Cross-Domain Application: Examining the applicability and effectiveness of the proposed techniques in diverse domains such as finance, healthcare, and environmental science could reveal insights into their versatility and adaptability to various types of regression problems.
5. Enhancing Autoencoder Architectures: Future studies might focus on optimizing the architecture of Target Aware Autoencoders and AutoInflaters, including exploring different neural network models and training procedures to further improve their accuracy in estimating target values.

By pursuing these future directions, the research community can build upon the foundation laid by this paper, driving forward the development of robust solutions to the persistent challenge of imbalanced data in regression analysis.

Code availability

The code base is available at https://github.com/ashhadulislam/augmentdatalib_reg_source/tree/main And the python package can be installed from https://pypi.org/project/knnor-reg/ The following Jupyter notebook contains an end-to-end example implementation of the algorithm. https://github.com/ashhadulislam/augmentdatalib_reg_source/blob/main/example/Example.ipynb. This information has been added to the conclusion section (Section 5) in the paper.

Declaration of competing interest

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

Data availability

Open source Links to codebase has been mentioned in the paper.

Acknowledgments

Open Access funding provided by the Qatar National Library.

References

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

Ali, M. (2020). PyCaret: An open source, low-code machine learning library in Python. PyCaret version 1.0.

Barupal, D. K., & Fiehn, O. (2019). Generating the blood exposome database using a comprehensive text mining and database fusion approach. *Environmental Health Perspectives*, *127*(9), 2825–2830.

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, *49*(2), 1–50.

Branco, P., Torgo, L., & Ribeiro, R. P. (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 Systems with Applications*, Article 116387.

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

Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2015). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*.

dos Santos Coelho, L., Hultmann Ayala, H. V., & Cocco Mariani, V. (2024). CO and NOx emissions prediction in gas turbine using a novel modeling pipeline based on the combination of deep forest regressor and feature engineering. *Fuel*, *355*, Article 129366.

Douzas, G., & Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. *Information Sciences*, *501*, 118–135.

Elhassan, T., & Aljurf, M. (2016). Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. *Global Journal of Technolology and Optimization S*, *1*, 2016.

Elor, Y., & Averbuch-Elor, H. (2022). To SMOTE, or not to SMOTE? arXiv preprint arXiv:2201.08528.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets*: *vol. 10*, Springer.

Gan, D., Shen, J., An, B., Xu, M., & Liu, N. (2020). Integrating TANBN with cost sensitive classification algorithm for imbalanced data in medical diagnosis. *Computers & Industrial Engineering*, *140*, Article 106266.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Islam, A., & Belhaouari, S. B. (2021). Class aware auto encoders for better feature extraction. In *3rd International conference on electrical, communication, and computer engineering* (pp. 1–5). IEEE.

Islam, A., Belhaouari, S. B., Rehman, A. U., & Bensmail, H. (2022a). K nearest neighbor OveRsampling approach: An open source python package for data augmentation. *Software Impacts*, *12*, Article 100272.

Islam, A., Belhaouari, S. B., Rehman, A. U., & Bensmail, H. (2022b). KNNOR: An oversampling technique for imbalanced datasets. *Applied Soft Computing*, *115*, Article 108288.

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 1–54.

Juez-Gil, M., Arnaiz-González, Á., Rodríguez, J. J., & García-Osorio, C. (2021). Experimental evaluation of ensemble classifiers for imbalance in big data. *Applied Soft Computing*, *108*, Article 107447.

Kohler, M., & Langer, S. (2021). On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, *49*(4), 2231–2249.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *International conference on machine learning*: *vol. 97*, (pp. 179–186). Morgan Kaufmann.

Laza, R., Pavón, R., Reboiro-Jato, M., & Fdez-Riverola, F. (2011). Evaluating the effect of unbalanced data in biomedical document classification. *Journal of Integrative Bioinformatics*, *8*(3), 105–117.

Liu, N., Shen, J., Xu, M., Gan, D., Qi, E.-S., & Gao, B. (2018). Improved cost-sensitive support vector machine classifier for breast cancer diagnosis. *Mathematical Problems in Engineering*, *2018*, 1–13.

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., & Zafeiriou, S. (2017). Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 51–59).

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*, 21.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on international conference on machine learning* (pp. 833–840). Omni Press.

Rothe, R., Timofte, R., & Van Gool, L. (2018). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, *126*(2–4), 144–157.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression. *eScholarship*.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687–719.

Thanathamathee, P., & Lursinsap, C. (2013). Handling imbalanced data sets with synthetic boundary data generation using bootstrap re-sampling and AdaBoost techniques. *Pattern Recognition Letters*, *34*(12), 1339–1347.

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

Torgo, L., & Ribeiro, R. (2007). Utility-based regression. In *PKDD 2007*: *11th European conference on principles and practice of knowledge discovery in databases*: *vol. 7*, (pp. 597–604). Springer.

Torgo, L., Ribeiro, R. P., Pfahringer, B., & Branco, P. (2013). Smote for regression. In *Progress in artificial intelligence: 16th portuguese conference on artificial intelligence* (pp. 378–389). Springer.

Tunçay, T., Alaboz, P., Dengiz, O., & Başkan, O. g. (2023). Application of regression kriging and machine learning methods to estimate soil moisture constants in a semi-arid terrestrial area. *Computers and Electronics in Agriculture*, *212*, Article 108118.

Vapnik, V., & Vapnik, V. (1998). Statistical learning theory wiley. *New York*, *1*(624), 2.

Wang, Y., Yao, H., & Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomputing*, *184*, 232–242.

Yang, Y., Zha, K., Chen, Y., Wang, H., & Katabi, D. (2021). Delving into deep imbalanced regression. In *Proceedings of the 38th international conference on machine learning* (pp. 11842–11851). MLR Press.

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 2528–2535). IEEE.

Zhong, J., He, Z., Guan, K., & Jiang, T. (2023). Investigation on regression model for the force of small punch test using machine learning. *International Journal of Pressure Vessels and Piping*, *206*, Article 105031.