Journal of Artificial Intelligence Research 82 (2025) 111-136 Submitted 02/2024; published 01/2025

A Selective Under-Sampling (SUS) Method For Imbalanced Regression

Jovana Aleksic JOVANA.R.ALEKSIC@GMAIL.COM
Universidad Politécnica de Madrid, Spain
Weill Cornell Medicine in Qatar, Qatar

Miguel García-Remesal MGREMESAL@FI.UPM.ES
Biomedical Informatics Group,
Departamento de Inteligencia Artificial,
Universidad Politécnica de Madrid, Spain

Abstract

Many mainstream machine learning approaches, such as neural networks, are not well suited to work with imbalanced data. Yet, this problem is frequently present in many real-world data sets. Collection methods are imperfect, and often not able to capture enough data in a specific range of the target variable. Furthermore, in certain tasks data is inherently imbalanced with many more normal events than edge cases. This problem is well studied within the classification context. However, only several methods have been proposed to deal with regression tasks. In addition, the proposed methods often do not yield good performance with high-dimensional data, while imbalanced high-dimensional regression has scarcely been explored. In this paper we present a selective under-sampling (SUS) algorithm for dealing with imbalanced regression and its iterative version SUSiter. We assessed this method on 15 regression data sets from different imbalanced domains, 5 synthetic high-dimensional imbalanced data sets and 2 more complex imbalanced age estimation image data sets. Our results suggest that SUS and SUSiter typically outperform other state-of-the-art techniques like SMOGN, or random under-sampling, when used with neural networks as learners.

1. Introduction

The increasing availability of large data sets over the last few years enables researchers to use machine learning methods for building practical predictive models. However, such data availability comes at the cost of less-than-perfect data with imbalanced or skewed distributions. In regression tasks uniform target distributions normally improve the performance of different machine learning algorithms like neural networks (Fernández et al., 2018). Nevertheless, real-world data sets often contain target values in certain ranges occurring significantly less frequently than in other dominant ranges. Consequently, models tend to perform better on rich information target regions than on scarce regions (Fernández et al., 2018). However, the latter are frequently more significant for building predictive models than other densely populated regions.

This problem has been widely studied within a classification context, but very rarely in a regression context. Existing approaches for classification tasks can be divided into two main groups: resampling techniques, and cost-sensitive learning approaches (Krawczyk, 2016). Adapting these techniques to regression problems is problematic due to fundamental differences between nominal and continuous target values. Ribeiro (2011) proposed a relevance function that splits the data set into rare and major classes, which enables using some of the existing imbalanced classification methods in regression problems. Only a few methods have been proposed to deal with regression problems with imbalanced data sets. These include SMOGN (Branco et al., 2017) which is the state-of-the-art algorithm for dealing with imbalanced regression. It is an improvement over SMOTER (Torgo et al., 2013) and combines it with oversampling via Gaussian noise. Random oversampling is another common technique used for imbalanced regression, but can lead to over-fitting (He & Garcia, 2009). Random under-sampling is another available approach but we consider it a blind loss of potentially useful information. Some samples are more valuable than others for learning algorithm's ability to predict target values accurately. In the classification setting, these could be samples close to the class edges. In a regression setting, these could be samples in feature space regions where small differences can cause large changes of the target value. Nevertheless, random under-sampling has shown better performance within high-dimensional imbalanced classification data setting (Blagus & Lusa, 2013), while SMOTE (Chawla et al., 2002) has been shown to introduce bias and perform poorly in this specific setting (Blagus & Lusa, 2013).

Therefore, in this paper we propose a new method to address the problem of imbalanced regression. This method, which we named SUS, utilizes a selective under-sampling strategy to extract the most representative samples from the data set. Inspired by information gain theory (Mitchell, 1997) we investigate both feature and target spaces in order to optimize the selection of samples that are the most significant information carriers. In addition, we propose a variant of the proposed method called SUSiter which is especially suited for iterative algorithms like neural networks. SUSiter exploits all available data in a training process while using only a selected subset of samples in each iteration. SUSiter exploits the advantages of using a balanced data set for building the regression model while enabling the use of a different subset of data samples in each learning iteration. This reduces the loss of available data samples to a minimum.

This paper is organized as follows: Section 2 defines the problem of imbalanced regression and presents an overview of the existing related work. The proposed SUS and SUSiter algorithms are described in Section 3, while the results of the experimental evaluation are presented in Section 4. We provide a discussion on the obtained results in Section 5. Finally, Section 6 outlines the main conclusions and future work.

2. Related Work

Imbalanced regression problems are a subset of regression problems with a non-uniform distribution of the target range where certain target ranges are significantly less represented than others. The goal is to obtain a model that approximates a function $Y = f(x)$. A training set $D = \{(x_i, y_i)\}_{i=1}^{N}$ with N samples is used. This problem is significantly more complex than imbalanced classification since the target variable has a potentially infinite number of values compared to the limited number of classes in classification problems.

Most of the existing approaches for imbalanced regression build on top of the Torgo and Ribeiro (2007) and Ribeiro's (2011) proposal of a relevance function. The relevance function $\phi(y)$ assigns a quantitative relevance score to the range of target values, where 1 corresponds to the maximal relevance and 0 to the minimum relevance. Nevertheless, the quantification of the relevance is domain-specific and ideally should be performed by domain experts which are not always available. To address this issue, Ribeiro (2011) proposed an automated method for approximating the relevance function $\phi(y)$. The latter is estimated from the target variable distribution through box plot statistics. The method assumes that the rare and most extreme cases are the most relevant. The relevance function is then used to classify data samples into major (normal) and minor (rare) using a user-provided threshold tr. Given this threshold, the data set is split into a set of rare samples RS, and a set of normal samples, NS as follows: $RS = \{(x, y) \in D : \phi(y) \geq tr\}$ and $NS = \{(x, y) \in D : \phi(y) < tr\}$.

In general, approaches dealing with imbalanced data set are divided into resampling methods and/or cost-sensitive learning methods (Krawczyk, 2016). Resampling methods modify the data sets directly and most frequently act before the learning algorithms. These modifications include over-sampling and/or under-sampling of a data set with the intention to balance the distribution. On the other hand, cost-sensitive methods modify existing learning algorithms to better handle non-uniform data distribution.
There are different approaches to address data imbalance for classification problems. Resampling methods for classification often create new samples for rare classes (over-sampling) and/or remove samples from common classes (under-sampling). SMOTE (Chawla et al., 2002) is regarded as the state-of-the-art resampling algorithm for classification. Kernel density estimator is used to estimate the feature distribution of minority classes (Kamalov, 2020). New minority class samples are generated using the estimated feature distribution. In cost-sensitive methods the loss of samples with rare classes is emphasized in the overall loss (Cui et al., 2019).

Unlike classification, regression is not so frequently studied in the context of imbalanced settings. Nevertheless, the development of methods that address imbalanced regression is crucial for real-world applications. These include, for instance, the statistical downscaling of precipitation described by Vandal et al (2017) and improved by methods proposed by Steininger et al (2021). There also exist a few different sampling approaches for imbalanced regression, which are applied during data pre-processing, such as SMOTER (Torgo et al., 2013). The latter is based on the original SMOTE method for classification (Chawla et al., 2002) and combines under-sampling of common data samples with over-sampling of rare cases, in order to create a more balanced distribution. The authors adapted SMOTE for use in regression domains by binning data samples into rare and normal partitions using the above-mentioned relevance threshold tr and relevance function $\phi(y)$. Data samples marked as relevant RS are over-sampled, thus creating new synthetic cases via interpolation of features and target values between two relevant data samples. On the other hand, non-relevant data NS samples are under-sampled. The SMOGN (Branco et al., 2017) algorithm builds on top of SMOTER and combines it with oversampling via Gaussian noise. Normally distributed noise is added to the features and the target value of rare data samples, therefore creating additional, slightly altered replicas of existing samples (Branco et al., 2016). Rare data samples are identified using the same method, i.e. relevance function, as SMOTER. SMOGN iterates over all rare samples and chooses between SMOTER's interpolation-based oversampling and Gaussian noise-based oversampling depending on the distance to the k-nearest neighbors. For small distances, SMOTER's interpolation is applied, since interpolation is deemed more reliable for close samples. Other rare data samples are over-sampled with Gaussian noise, while common data samples are randomly under-sampled. The authors report improvements compared to SmoteR (Torgo et al., 2013). Given the unavailability of other methods addressing the same problem, SMOGN can be considered the state-of-the-art among resampling techniques.

Not many cost-sensitive approaches for imbalanced regression exist. DenseLoss (Steininger et al., 2021) is an example of such method, and it is based on a density-based weighting scheme DenseWeight also presented in the same manuscript. Unlike previous work on weighting data samples based on the target value distribution, DenseWeight does not make any assumptions about which cases are rare since it determines relative rarity with a density function. Conversly to SMOTER (Torgo et al., 2013) and SMOGN (Branco et al., 2017), DenseLoss (Steininger et al., 2021) does not explicitly change the data set, by creating new samples or removing the existing ones.

3. Methods

In this section we introduce SUS, our proposed selective under-sampling approach for imbalanced data in regression tasks. In addition, we present and explain SUSiter, a variant of the SUS method especially tailored for iterative learning algorithms. SUSiter exploits all available data in an iterative fashion, while only using a selected subset of data in each iteration.

3.1 Selective Under-Sampling SUS

This algorithm is applied to data as a pre-processing step for tackling imbalanced domains and acts before the learning process stage. First, data is split into rare RS and normal samples NS. Random under-sampling approaches remove samples from a data set without assessment of the importance of each specific sample. We believe this is not a suitable method to perform under-sampling as some samples contribute more than others to the learning algorithm's ability to correctly estimate the function $f(x)$. SUS performs selective under-sampling of NS by removing some of the less valuable samples while keeping those that carry the most relevant information. SUS preserves all data samples from RS. This leads to a more balanced data set.

The main idea of SUS is to find and extract the subset of NS data samples that better characterizes both feature and target space. Figure 1 shows how SUS chooses samples in a toy two dimensional feature data set (upper part of the Figure 1). Red squares and blue stars represent samples that are chosen by the SUS algorithm. Red (square) samples would be selected by SUS since they are isolated in the corresponding region of the feature space. On the other hand, blue (stars) samples would be selected by SUS as clusters representatives in the feature space. The green (circled) cluster aggregates samples that present not only close feature space values, but also close target values (bottom part of the Figure 1). In addition, one of the samples belonging to the same cluster presents a target value which is significantly deviated. Note that this is just an example and further samples with divergent target values could also belong to the same cluster. In this scenario, SUS would choose a single sample as representative of the cluster samples, as well as all the cluster samples with skewed target values. Note also that the largest cluster in Figure 1 has more representatives than the other clusters. Larger clusters (this is defined by further explained user thresholds) are processed in the algorithm as multiple smaller clusters. This ensures that in case large proportion of data samples is allocated in a large cluster SUS does not select only one sample as the cluster's representation.

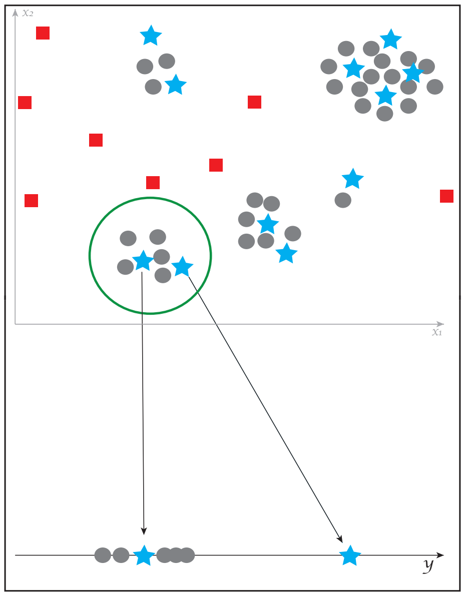


Figure 1: Toy example of two-dimensional feature space representing a selection of samples (red-squares and blue-stars). SUS selection and target value spread of the green cluster samples.

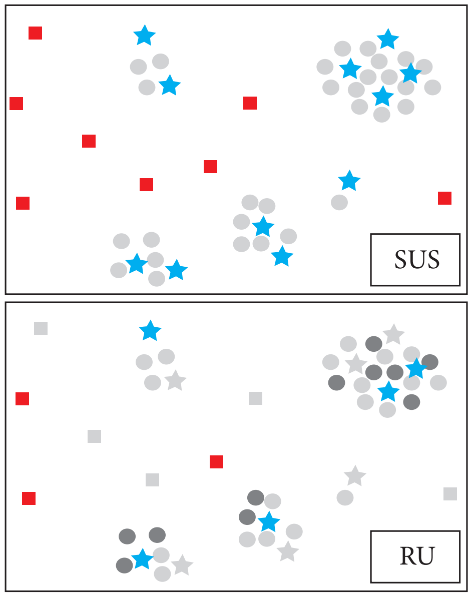


Figure 2: Toy example of two-dimensional feature space representing a selection of samples (red-squares, blue-stars and dark-grey-circles). SUS vs random under-sampling (RU).

To summarize, SUS firstly chooses every isolated sample from the feature space. We believe that such samples carry important information about the feature space representation which, if removed, would be lost. Therefore, the first step in our selection is based on the feature representation space: SUS selects samples without close neighbors. The second step is targeted at identifying dense sample clusters and their representatives. Finally, the third step is aimed at processing the discovered clusters contemplating that we are dealing with a regression problem instead of a classification one. In this regard, we need to consider not only the feature space, but also the target space. The clusters obtained in the previous stage are the best source for less-harmful removal of samples. Most of the cluster samples often can be discarded since the information they carry at feature level is already provided by the cluster representative. However, as stated earlier, SUS also considers the information that each sample carries at target level to decide whether to discard it or not. To assess the relevance of the target-level information carried by the samples belonging to the cluster, we determine how spread are the target values belonging to these samples. In case target values are widely spread across the target range, there is a potential for information loss with the removal of some of the samples from the cluster. In this case, SUS chooses the sample with the most-distant target value from the cluster's average and assesses the spread again. The idea is that the sample presenting the most distant target value is significantly different than others in the cluster. This process is iterated until a predefined spread threshold is reached.

Figure 2 shows an example of the difference between randomly under-sampled data and SUS selection with the same number of selected (highlighted by brighter colors/different shapes) samples. It can be noted that random under-sampling removes some of the relevant samples in the feature space, unlike SUS.

The SUS algorithm requires D: data set with continuous target variable y, tr: threshold for the relevance function (Ribeiro, 2011) for separation of D into RS and NS, k: number of neighbors in knn model, $blob_{tr}$: a threshold to decide which neighbors are considered close and which neighbours form part of the cluster, and $spread_{tr}$: a threshold for the dispersion of target values in clusters. Note that $blob_{tr}$ is not an absolute value in distance metrics, under which a neighbor is considered close. This would be difficult to provide without any prior knowledge of the data set and relations between samples in a data set. Therefore, we propose an automated method for determining this distance for every data set. $blob_{tr}$ is a percentile of the average distance to k neighbors, for the samples in NS. In Algorithm 1 we show how to compute $blob$ - a corresponding distance which is an actual threshold for the closeness of neighbors. $spread_{tr}$ is a dispersion threshold used in cluster processing for terminating the selection of cluster representatives. Once the dispersion of the target values is below $spread_{tr}$ the algorithm selects only the sample that has the target value the closest to the cluster's average.

The details of the SUS method are provided in Algorithm 1.

The SUS algorithm returns a reduced data set - with fewer samples than the original - and a more balanced target distribution. The returned data set contains all samples marked as rare RS in the original data set and the selected subset of NS values marked as normal in the original data set.
In Section 4 we provide further analysis of the hyper-parameters the in SUS algorithm.

3.2 SUS with Iterative Replacement - SUSiter

The algorithm is an adaptation of the previously proposed SUS which is especially tailored for iterative learning methods. Even though the SUS algorithm strives for the least impactful data loss, we still lose some information carried by the removed samples. With SUSiter method, we introduce a variant that allows for the use of all available data while retaining the under-sampling advantages. With this modification, SUSiter introduces a moderated amount of noise in the learning process which can help with the prevention of over-fitting (Goodfellow et al., 2016).

The main idea of SUSiter adaptation is an exploitation of available data samples. The process is roughly the same as the one previously described with SUS. The most important distinction is that SUSiter acts both before and during the learning process stage. The initial pre-processing before the learning process stage is the same as in the SUS. However, during the learning process, in each iteration data selected from feature space clusters is randomly replaced. This leads to a subset of data that is different in every iteration - thereby introducing the noise in the learning process - while keeping the central idea of

Algorithm 1 SUS

```
Require: $D$ - data set with target continuous variable $y$
        $tr$ - threshold for relevance on $y$ values
        $k$ - number of neighbours in $knn$ model
        $blob_{tr}$ - percentile threshold for the close neighbors distance    ▷ Default value 75%
        $spread_{tr}$ - threshold for the cluster target values dispersion    ▷ Default value 0.5
  procedure SUS($D$)
    Split $D$ into $NS$ and $RS$    ▷ (Ribeiro, 2011)
    Construct $knn(k, NS)$ model    ▷ (Cover & Hart, 1967)
    for $s \in NS$ do    ▷ Method for close neighbors threshold
      $kn \leftarrow \{(s_i, d_i) : s_i \in NS, min(d_i)\}_1^k$    ▷ $k$ neighbors from $knn$ model
      $d_s \leftarrow \frac{\sum_1^k d_i}{k}$    ▷ Average distance to $k$ neighbors
      $s.visited \leftarrow$ False    ▷ Initialization
    end for
    $d_{array} \leftarrow [d_{s1}, d_{s2}...d_{sN}]$, $d_{s(i)} < d_{s(i+1)}$    ▷ Sorted array of average distances to $kn$
    $blob \leftarrow blob_{tr}$-th percentile of $d_{array}$    ▷ $blob$ threshold for close neighbours
    for $s \in NS$ do    ▷ Under-sampling procedure
      if s.visited == False then    ▷ Visit every sample once
        $kn \leftarrow \{(s_i, d_i) : s_i \in NS, min(d_i)\}_1^k$    ▷ $kn$ set of $k$ closest neighbors
        if $\forall$ $(s_i, d_i) \in kn$, $d_i > blob$ then    ▷ No close neighbours
          $select\ s$, $s.grade \leftarrow$ II    ▷ Isolated sample, Grade II the highest level
          for $s_i \in kn$ do
            $s_i$.visited $\leftarrow$ True
          end for
        else    ▷ Cluster with close neighbors
          $cn \leftarrow \{s_i \in kn : d_i < blob\}_i^c$    ▷ Determine close neighbors set $cn$
          $y_{cn} \leftarrow \{y_i\}_1^c$ target values $\forall$ $s_i \in cn$    ▷ Target values for $cn$
          $\bar{y_{cn}} \leftarrow \frac{\sum_1^c y_i}{c}$    ▷ Average of target values
          $var_y \leftarrow \frac{\sum_1^c (y_i - \bar{y_{cn}})^2}{c}$    ▷ Variance for target values
          $spread \leftarrow \frac{var_y}{\bar{y_{cn}}}$    ▷ Spread in target space
          while $spread > spread_{tr}$ do
            $select\ s$, $\{s \in cn : max(y_s - \bar{y_{cn}})\}$    ▷ $y_s$ corresponding $y$ for sample $s$
            $cn \leftarrow \{s_i \in kn, s_i \neq s : d_i < blob\}$    ▷ Update $cn$, remove $s$
            $\bar{y_{cn}} \leftarrow \frac{\sum_1^{|cn|} y_i}{|cn|}$
            $var_y \leftarrow \frac{\sum_1^{|cn|} (y_i - \bar{y_{cn}})^2}{|cn|}$
            $spread \leftarrow \frac{var_y}{\bar{y_{cn}}}$
          end while
          $select\ s$, $\{s \in cn : min(y_s - \bar{y_{cn}})\}$    ▷ The closest sample to $\bar{y_{cn}}$
          $s.grade \leftarrow$ I    ▷ Grade I, middle importance
          $\forall$ $s_i \in cn$, $s_i$.visited $\leftarrow$ True
        end if
      end if
    end for
  end procedure
```

SUS which is to have the best representation of both the feature and target spaces and a more balanced distribution of target values. It is important that the number of selected samples is the same in each iteration, and the same as in SUS, with some of the cluster representatives being randomly replaced with their close neighbors.

Algorithm 2 SUSiter

```
Require: $D$ - data set with target continuous variable $y$
    $tr$ - threshold for relevance on $y$ values
    $k$ - number of neighbours in $knn$ model
    $blob_{tr}$ - percentile threshold for the close neighbors distance    ▷ Default value 75%
    $spread_{tr}$ - threshold for the cluster target values dispersion    ▷ Default value 0.5

procedure SUSiter($D$)
    Perform SUS Algorithm 1                                     ▷ As a pre-processing step

    for $i$ in $iterations$ do                                  ▷ $i$ depends on the learning algorithm
        for $s \in NS$ do                                       ▷ Under-sampling procedure
            if $s.grade$ == II then                             ▷ $grade_s$ importance of a sample
                select $s$
            else
                if $s.grade$ == I then                          ▷ Sample $s$ has close neighbours
                    $cn \leftarrow \{s_i \in NS : d_i < blob\}_i^c$    ▷ close neighbors set $cn$
                    select random $s_i \in cn$                  ▷ Select random close neighbour
                end if
            end if
        end for
        Perform learning algorithm iteration                    ▷ In NNs one epoch
    end for
end procedure
```

3.3 Information Theory Analysis

Information theory is grounded in the fundamental idea that discovering an improbable event imparts more information than discovering a probable one (Goodfellow et al., 2016). In simpler terms, if an event is likely or certain, it carries less information, even to the point of having none in the case of events guaranteed to happen. Conversely, events with lower likelihood possess higher information content. (Goodfellow et al., 2016).

Translating this analogy to the problem of imbalanced regression, envisioning the n-dimensional feature space through the lens of a sample probability distribution (illustrated in Figure 3) reveals that regions corresponding to clustered areas possess a higher likelihood of occurrence, while the sparser spaces between clusters exhibit a lower probability. If we identify a subset of data samples with SUS (blue stars and red squares in Figure 3), incorporating the remaining samples (grey circles) back into the data set will predominantly occur in regions characterized by higher probability. Consequently, these additional samples may not contribute significantly to the information content, aligning precisely with the primary objective of the developed algorithm - the removal of low-informative data samples.

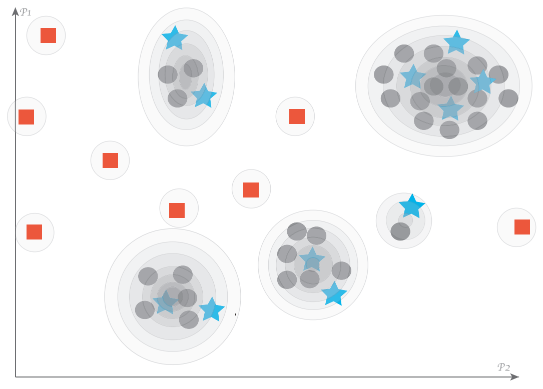


Figure 3: Toy example of two-dimensional feature space with a sample probability distribution.

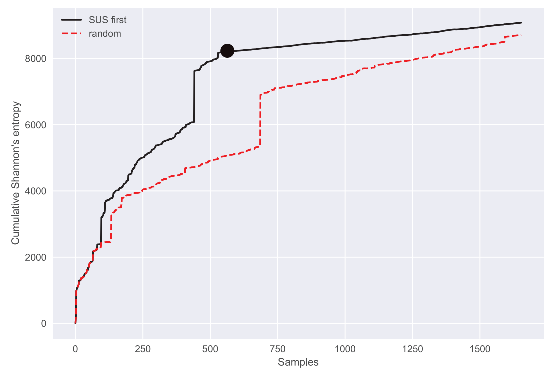


Figure 4: Entropy increase with the addition of new data samples to the empty data set.

This analogy further extends to the samples within clusters and their corresponding target values. Samples with similar target values are in the denser probability regions and therefore carry less information when compared to each other, especially when contrasted with samples with distinct target values.

Furthermore, we have undertaken an endeavor to demonstrate the effectiveness of SUS with information theory, employing the real imbalanced data set "Accel" (details of which will be provided in Section 4). The graphical representation in Figure 4 illustrates the cumulative entropy (information) as new data samples, sourced from the actual data set, are systematically introduced to an initially vacant data set.

To elaborate on the methodology, we initiate the process with an empty data set, whereby each new sample (from the real data set) is incrementally added. Prior to incorporation, a Gaussian kernel density estimation (Zhang & Karunamuni, 1998) of probability is executed to approximate the likelihood of the new data sample within a given region (with respect to the data samples already added to the data set). Subsequently, Shannon's entropy (Shannon, 1948) for the new data sample is computed. The cumulative entropy for all samples added up to that point is then plotted. Crucial to the understanding of the process is the sequence in which the data samples are introduced. The black (full) line represents a case where the samples chosen by SUS from the original data set are initially added, followed by the integration of the remaining samples. The dot on the graph marks the last data sample chosen by SUS. The red (dashed) line represents a case where samples are randomly added. It is worth noting the rapid increase of entropy in the beginning, followed by a tapering effect. This difference in the entropy increase is stronger in the case where SUS samples are initially introduced suggesting that the selected data subset is not random but rather optimized for entropy gain. It is also notable that there are no major entropy jumps after the dot in the black line, while there are some in the dashed line.

In conclusion, the underlying principle that governs the SUS technique is fundamentally rooted in the optimization of information concerning the number of selected samples. More precisely, SUS is designed to achieve the maximization of information capture within the most minimal sample size feasible, while avoiding the loss of entire regions within the feature space.

4. Experimental Evaluation

We designed an experimental setup targeted at assessing the performance of the SUS and SUSiter strategies in the context of imbalanced regression tasks.

4.1 Data

For evaluating the performance of the presented approaches, we used three different types of data sets - standard imbalanced data, synthetic high-dimensional imbalanced data, and more complex age estimation image data sets IMDB-WIKI (Rothe et al., 2018) and AgeDB (Moschoglou et al., 2017).
We selected 15 standard regression data sets from different imbalanced domains. Table 1 shows the main characteristics of these data sets. N represents the number of samples in a data set, f.total is the number of features, f.nom is the number of nominal features and f.num is the number of numeric predictors. nRare is the number of samples with relevance value, determined by Ribeiro (2011), higher than the threshold (0.8) and finally %Rare represents a percent of rare samples compared to the entire data set size. Figure 5 shows target value distributions for each data set in the standard data group.

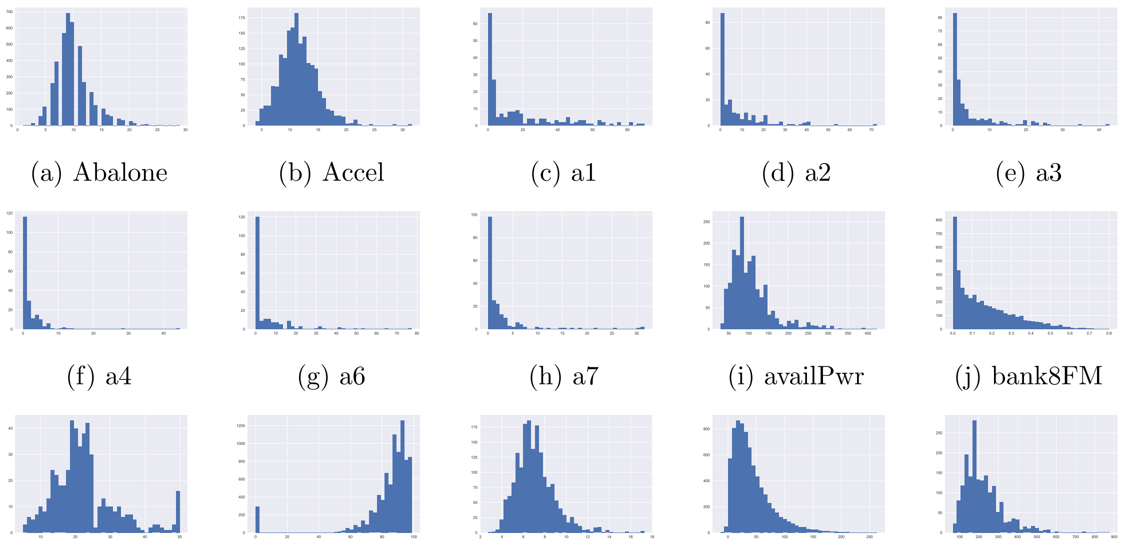
Furthermore, we created 5 synthetic imbalanced high-dimensional data since imbalanced regression represents a special challenge in high-dimensions. We use two different methods for generating the synthetic data. In the first method, the target value is generated by applying a random linear regression model to the previously generated input and a Gaussian-centered noise with an adjustable scale (make_regression) (Pedregosa et al., 2011). We also resort to a Multilayer Perceptron (MLP) as a random function to generate the remaining synthetic data sets. This assumes that the function can be learned again by an MLP. Our network's parameters are initialized with a standard Gaussian distribution. The features are also drawn from a standard Gaussian distribution. The network consists of 3 hidden layers (30, 10, 3 neurons per layer, respectively) and ReLU (Nair & Hinton, 2010) activation. The final hidden layer is connected to a single neuron with linear activation to obtain target values for a regression task. We designed the data sets to cover a wide range of sample and feature sizes, their ratios, the percentage of rare data and to have present one or two extremes.

Table 2 shows the main features of the generated synthetic high-dimensional data sets and the methods employed for their generation, while Figure 6 show target value distributions for each data set in the synthetic HD data group.

Lastly, to evaluate the efficacy of our proposed method on deep learning architectures, we have selected the more intricate age estimation image data sets, namely IMDB-WIKI (Rothe et al., 2018) and AgeDB (Moschoglou et al., 2017). To ensure data quality, we filtered

Table 1: Standard data sets information.

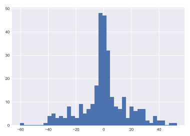
DataSet	N	f.total	f.nom	f.num	nRare	%Rare
Abalone	4177	8	1	7	679	16.3
Accel	1732	15	3	12	89	5.1
a1	198	11	3	8	28	14.1
a2	198	11	3	8	22	11.1
a3	198	11	3	8	32	16.2
a4	198	11	3	8	31	15.7
a5	198	11	3	8	21	10.6
a6	198	11	3	8	33	16.7
a7	198	11	3	8	27	13.6
availPwr	1802	16	7	9	157	8.7
bank8FM	4499	9	0	9	288	6.4
boston	506	13	0	13	65	12.8
cpuSm	8192	13	0	13	713	8.7
fuelCons	1764	38	12	26	164	9.3
heat	7400	11	3	8	664	8.9
maxTorque	1802	33	13	20	129	7.2

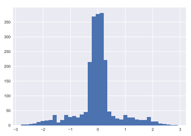

(a) Abalone (b) Accel (c) a1 (d) a2 (e) a3

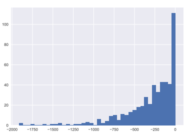
(f) a4 (g) a6 (h) a7 (i) availPwr (j) bank8FM

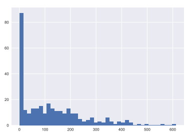
(k) boston (l) cpuSm (m) fuelCons (n) heat (o) maxTorque

Figure 5: Distributions of target values of standard data sets.


(a) synthHD_1


(b) synthHD_2


(c) synthHD_3


(d) synthHD_4

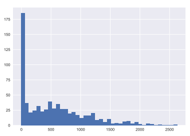

(e) synthHD_5

Figure 6: Distributions of target values of HD data sets.

Table 2: Synthetic high-dimensional data sets information.

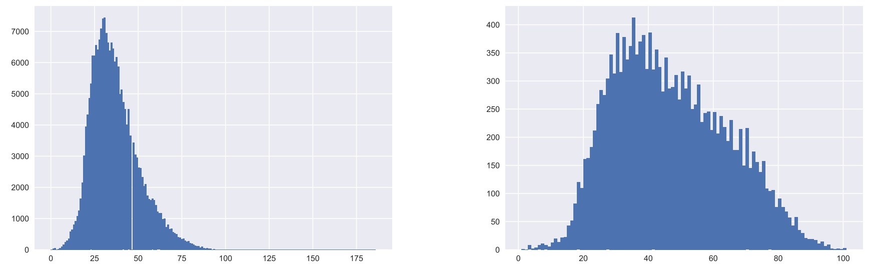
DataSet	N	f.total	f.nom	f.num	nRare	%Rare	Method
syntheticHD_1	293	1000	0	1000	82	27.9	make_regression
syntheticHD_2	2228	6000	0	6000	89	23.3	make_regression
syntheticHD_3	500	20000	0	20000	44	8.8	MLP
syntheticHD_4	300	15000	0	15000	22	7.3	MLP
syntheticHD_5	700	15000	0	15000	37	5.2	MLP

out inadequately scored images and preprocessed them to be square-shaped with uniform dimensions, as described in (Yang et al., 2021) and further elucidated in the Appendix. Table 3 shows the main features of these data sets. im.dim represents a dimension of images once processed. Since the sizes of these data sets are more significant than the previous ones columns test.size and val.size show the corresponding test/validation number of samples. Figure 7 show the age distribution in these data sets. The test and validation data is *balanced* as well.

For all experiments, we obtained a relevance function for each data set through the automated method proposed by Ribeiro 2011 (Python implementation by Kunz 2020). In this method, the quartiles and interquartile range of the target variable distribution are used for assigning a higher relevance to both high and low extreme values of the target variable. Therefore, the considered data sets will have either one extreme (on the high or low values of the target variable) or two extremes (high and low extremes of the target variable). We consider a threshold of 0.8 on the relevance values in all data sets to obtain the set of rare samples, as reported in SMOGN paper (Branco et al., 2017).

Table 3: Image data sets information.

DataSet	N	im.dim	nRare	%Rare	test.size	val.size
IMDB-WIKI	213553	224×224	17315	8.1	11022	11022
AgeDB	16488	224×224	293	1.8	2140	2140



(a) IMDB-WIKI (b) AgeDB

Figure 7: Distributions of target values for image data sets.

Important to note is that we selected the *balanced* test data set which implies that target values of the test data set are seeded uniformly throughout the whole target range. Random sampling from a data set would create imbalanced test data and consequently cause a bias towards a more abundant target value region in model performance assessment. Selected test data covers 20% of the whole corresponding data set and is completely held away from the training process. All features were normalized before the learning algorithm stage. We repeated every setting 30 times, shuffled random samples within data sets to avoid any effects introduced by the specific order of samples, and used averages of mean squared error (MSE) to report an evaluation metric.

4.2 Resampling Techniques

We applied to each of the 15 standard regression problems, 5 synthetic high-dimensional data sets and 2 age estimation image data sets, 5 different resampling strategies. The techniques that we tested are as follows:

- original data set (without any resampling)
- random under-sampling RU
- SMOGN algorithm
- SUS algorithm
- SUSiter algorithm

4.3 Hyper-Parameter Analysis

Prior to presenting the results, it is imperative to highlight the hyper-parameters that exert influence on the SUS algorithm. The foremost among these is the number of neighbors in the *knn* model (Cover & Hart, 1967), denoted as k. Larger values of k facilitate the detection of more expansive clusters. For instance, setting $k = 3$ implies that each cluster accommodates a maximum of 4 samples, with at least one serving as the representative. In the case of $k = 10$, the cluster accommodates a maximum of 11 samples, potentially

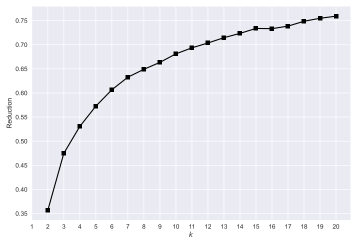


Figure 8: Reduction of NS depending on k parameter value.

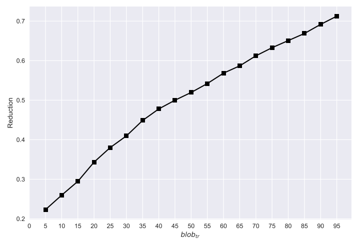


Figure 9: Reduction of NS depending on $blob_{tr}$ threshold.

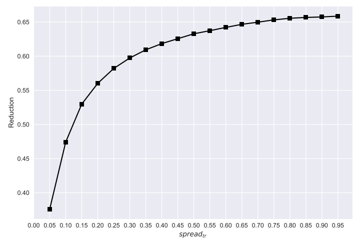


Figure 10: Reduction of NS depending on $spread_{tr}$ threshold.

featuring only one representative. Consequently, larger values of k result in a more robust reduction of samples. Figure 8 shows the reduction in the size of the Abalone data set contingent upon the varying values of k in the knn model of the SUS algorithm. The focus is specifically on the percentage reduction for the NS part of the dataset, while the rare samples RS remain untouched for the learning stage. Notably, the percentage of reduction plateaus around 74%. Although the general curve shape remains consistent across all data sets, specific values fluctuate based on the unique characteristics of the data. All other parameters in the algorithm are held constant.

Another critical hyper-parameter is the $blob_{tr}$ size, which, akin to the previous case, influences the consideration of more neighbors as close. This, in turn, results in the detection of larger clusters, contributing to a more pronounced reduction in the size of the data set, specifically the NS part. Figure 9 elucidates the impact of varying the $blob_{tr}$ parameter on the strength of reduction. Similar to the preceding graph, the focus is solely on the effect on the NS samples, while the rare samples remain unaltered. All other parameters are maintained at a constant value, with $k = 7$ neighbors in the knn model.

The third hyper-parameter, $spread_{tr}$, assumes significance as larger values allow more spread in target values within clusters. This facilitates the representation of only one sample as a representative of a larger group, consequently augmenting the reduction in the data set's size. Figure 10 shows how the reduction varies with changes in $spread_{tr}$ parameters. The data set used for demonstration remains consistent with the previous two figures (Abalone), employing a k parameter of 7 and a $blob_{tr}$ of 75%.

Acknowledging the nature and rationale of the SUS algorithm, we conduct an analysis of all hyper-parameters. However, it is evident from the preceding explanation that $blob_{tr}$ and $spread_{tr}$ primarily influence the algorithm by modulating the strength of reduction in the NS part of the data set, a parameter also controlled by k. Consequently, in our experiments, we systematically evaluate three distinct k parameter values (5, 7, and 10), maintaining $blob_{tr} =$ 75% and $spread_{tr} = 0.5$ as constants. These specific values are established through rigorous experimental evaluation, consistently yielding optimal results. As a result, we advocate treating them as constants and recommend their adoption as default options. Owing to their nuanced interplay with the k parameter, we refrain from presenting evaluations for alternative $blob_{tr}$ and $spread_{tr}$ values, as their effects are inherently encapsulated by the k parameter.

4.4 Learning Method

We tested our algorithm on neural networks (NNs). Most of the recent major advances in machine learning (ML) are achieved using advanced NNs, and therefore we consider NNs the most promising learning algorithm currently existing in the ML community (Jumper et al., 2021), (Brown et al., 2020), (Carion et al., 2020), (Goh et al., 2021). However, a prerequisite for NNs to perform well is that the data is approximately balanced (Castro & Braga, 2013), (Wang et al., 2016). Furthermore, authors of SMOGN, the state-of-the-art algorithm, state that their algorithm displays the highest advantages when used with random forests, and multivariate regression splines, and has a smaller impact when used with neural networks (Branco et al., 2017). Therefore, we designed SUS and especially SUSiter with a focus on NNs performance. We test 4 different architectures for the standard data sets and synthetic high-dimensional data (deeper, shallower, wider and narrower) in order to show that performance is not architecture-specific: (32, 16, 8), (8, 4, 2), (20, 5), (10, 5). The specific values have shown to cause convergence of all models for all data sets (within standard and synthetic high-dimensional data) and all sampling methods. Training is performed in the same way for all data sets to avoid bias potentially introduced by different training times or optimization methods. Moreover, for the two age estimation data sets we use the ResNet architecture. Details of the training parameters for all data sets are described in Appendix. In total, we evaluated 890 combinations.

4.5 Results

We present here the performance of SUS and SUSiter on the data sets we performed our experiments on. Figures 11 and 13 summarize the most important results, for standard and synthetic high-dimensional data sets. Values in the pie charts present a number of the data sets belonging to the specific sampling strategy that performs the best with the data set type, with root-mean-squared error (RMSE) being the evaluation metric. In these figures only one setting $k = 7$ in SUS and SUSiter is considered. We use only one setting for reporting the main result as it is fair taking into account that we use only one setting for SMOGN, and a corresponding reduction factor for RU, even though better settings for specific data sets might exist. The reported numbers are summed across all 4 neural network architectures.

Figure 12 and 14 report the number of the best-performing techniques for any setting of $k \in \{5, 7, 10\}$ in *knn* model for SUS techniques and corresponding RU factors, for standard and synthetic high-dimensional data sets, respectively.

Since we test different architectures of the NNs, Table 4 and 5 show more detailed reports which include information about the architectures and sampling strategies performance. Table 4 reports numbers when only one setting $k = 7$ is considered, while Table 5 reports results for any setting $k \in \{5, 7, 10\}$.

Given the increased complexity of the age estimation image data sets in terms of size and neural network configuration, we find it important to provide a comprehensive presentation of evaluation results, substantiated by specific numerical values. The detailed outcomes of this evaluation can be found in Table 6.

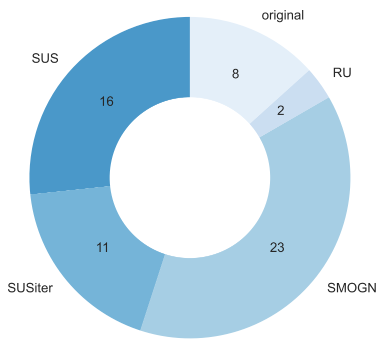


Figure 11: Pie chart of best performing strategies for standard data and $k = 7$ for SUS and SUSiter. RU factors are corresponding.

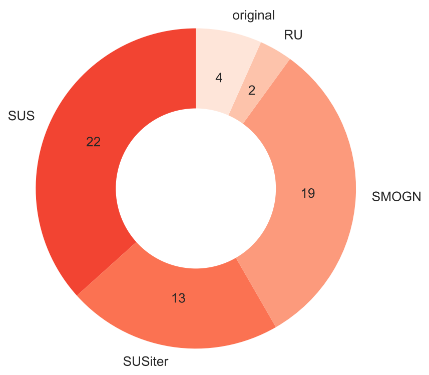


Figure 12: Pie chart of best performing strategies for standard data with any $k \in \{5, 7, 10\}$. RU factors are corresponding.

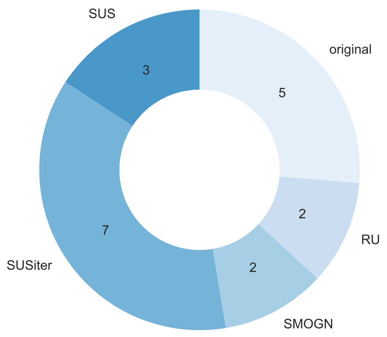


Figure 13: Pie chart of best performing strategies for synthetic HD data sets and $k = 7$ for SUS and SUSiter. RU factors are corresponding.

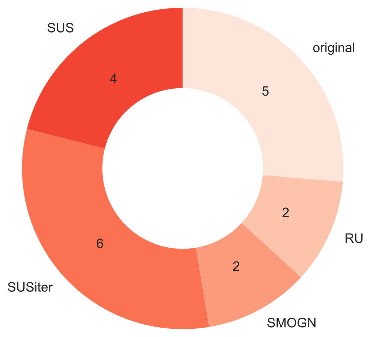


Figure 14: Pie chart of best performing strategies for synthetic HD data sets with any $k \in \{5, 7, 10\}$. RU factors are corresponding.

Table 4: Number of best-performing data sets per sampling technique and per neural network architecture. For SUS/SUSiter only one $k = 7$ is considered. AnySUS represents either SUS or SUSiter.

DataSetType	Architecture	SUS	SUSiter	AnySUS	SMOGN	RU	original
standard	(32, 16, 8)	3	4	7	5	1	2
	(8, 4, 2)	2	2	4	8	1	2
	(20, 5)	4	4	8	4	0	3
	(10, 5)	7	1	8	6	0	1
synthetic_HD	(32, 16, 8)	0	2	2	1	0	2
	(8, 4, 2)	2	1	3	0	1	1
	(20, 5)	0	2	2	1	0	2
	(10, 5)	1	2	3	0	1	1

Table 5: Number of best performing data sets per sampling technique and per neural network architecture. For SUS/SUSiter we consider $k \in \{5, 7, 10\}$. AnySUS represents either SUS or SUSiter.

DataSetType	Architecture	SUS	SUSiter	AnySUS	SMOGN	RU	original
standard	(32, 16, 8)	6	4	10	4	0	1
	(8, 4, 2)	6	2	8	5	1	1
	(20, 5)	4	5	9	4	1	1
	(10, 5)	6	2	8	6	0	1
synthetic_HD	(32, 16, 8)	0	2	2	1	0	2
	(8, 4, 2)	3	1	4	0	1	0
	(20, 5)	0	2	2	1	0	2
	(10, 5)	1	2	3	0	1	1

Table 6: Evaluation of the performance for the image data sets.

DataSet	Technique	MSE	$k = 5$	$k = 7$	$k = 10$
IMDB-WIKI	MSE	138.06			
	RU	134.12			
	SMOGN	136.09			
	SUS		134.70	131.98	132.52
	SUSiter		132.53	134.36	**131.53**
AgeDB	MSE	101.60			
	RU	106.05			
	SMOGN	117.29			
	SUS		101.70	**99.01**	99.87
	SUSiter		103.55	101.86	100.42

5. Discussion

In the previous section results of our experimental evaluation are presented. We have shown that SUS and SUSiter outperform the existing state-of-the-art approaches for imbalanced regression problems. However, there are different aspects to discuss.

As SUS acts only by removing some of the samples from the data set, random under-sampling is the closest existing method, and it is interesting to see how they compare. In our experiments, SUS outperformed RU approximately 60% of the time. The reduction of evaluation score is on average approximately 15%. For some data sets the difference in evaluation scores is especially large. For example, the fuelCons data set trained on a neural network with the (20, 5) architecture yields almost 4 times better results if pre-processing was done with SUS than with RU. The reason could be that some data sets include many isolated samples in feature space along with larger clusters of data, which causes RU to remove proportionally many outlying samples which are important for the algorithm's learning ability. Furthermore, random under-sampling has a much wider distribution of evaluation scores compared to SUS as can be seen in Figure 15. This example is with Accel data set, $k = 7$ in *knn* model for SUS sampling technique, corresponding factor is chosen for RU and (20, 5) architecture of the neural network. Due to inherently random process, some sub-selections of samples can be very detrimental to the model performance. With availPwr data set, and the same NN architecture as already described, approximately one out of 20 runs with RU is an outlier, with at least 10 times higher evaluation score.

Another aspect to discuss is how SUS and SUSiter compare. Even though SUSiter exploits more of the available data, SUS seems to work a bit better in practice for standard data sets (Figures 11 and 12). However, SUS has better evaluation scores than SUSiter only 50% of the time. A possible explanation lies in the fact that when SUS outperforms SUSiter it does so by improving the evaluation score by 15% on average, but the reverse is only true by 12%. The distribution of evaluation scores is typically very similar for these two methods, as shown in Figure 16. Parameters used in the figure are $k = 7$, Accel data set and (20,5) neural network architecture. Nevertheless, SUSiter performs better than SUS

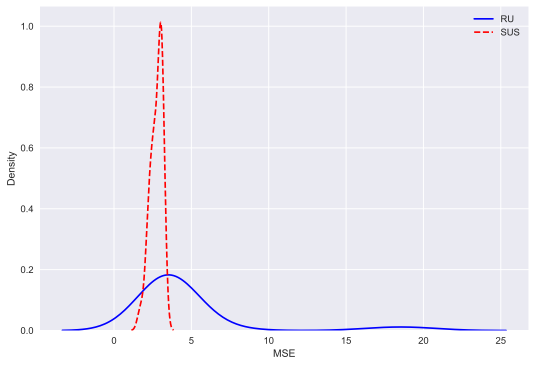


Figure 15: RU vs SUS. Distribution of evaluation scores, for Accel data set, $k = 7$ and (20, 5) neural network architecture.

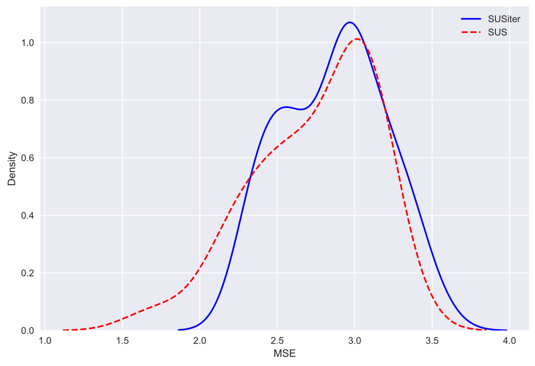


Figure 16: SUS vs SUSiter. Distribution of evaluation scores, for Accel data set, $k = 7$ and (20, 5) neural network architecture.

for high-dimensional setting, as can be seen in Figures 13 and 14. As mentioned earlier, this technique reduces the chances for overfitting to the training data set, which is especially important in high-dimensional settings. Furthermore, parameter k also affects the relationship between SUS and SUSiter. We note that depends on k which algorithm performs better. One setting of k can cause SUS to be better, while the other can cause SUSiter to be better. Finally, the two image data sets show that both SUS and SUSiter improve the baseline performance and outperform other techniques like RU or SMOGN, with the main differences coming from k parameter. Best performing values of SUS and SUSiter are very close, with SUS winning one data set and SUSiter winning the other.

The extent to which SUS reduces the size of the data set is dependent upon the inherent characteristics of the data. As illustrated in Figure 17, standard data set sizes after the SUS selection are depicted across different values of the parameter k. It is noteworthy that smaller data sets exhibit a considerably higher percentage of data samples selected by SUS. Larger data sets typically encompass regions of elevated density within the feature space, consequently resulting in a more pronounced reduction by SUS in comparison to smaller data sets, as exemplified by datasets a1-a7. Also, as discussed earlier in subsection 4.3, increasing the values of k tends to make the reduction more pronounced.

Execution speed is another aspect to be taken into account. SUS has shown to perform from 2 up to 30 times faster than SMOGN, depending on the data set which is used for benchmarking. The difference between execution speeds is the most significant, approximately 30 times, for larger data sets. SUSiter has shown to perform at approximately the same speed as SMOGN. We compare speeds on a desktop iMac 2017 machine with 4 cores, and 3.5 GHz Quad-Core Intel Core i5 processor.
Computation of nearest neighbors can be computationally intensive for high-volume data in practical settings. Nevertheless, the computation of knn tensors is parallelizable, since the computations are independent of each other, which is a natural fit for GPU computation (Li & Amenta, 2015). Furthermore, many methods addressing this challenge in a distributed

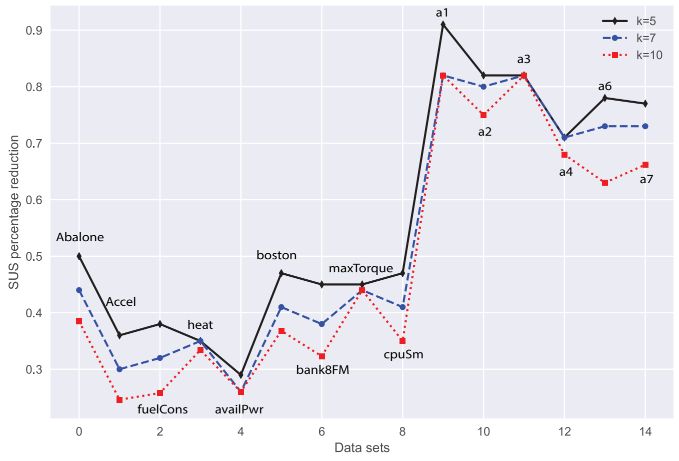


Figure 17: Size of standard data sets after SUS selection for different k values.

way have been proposed (Zhang et al., 2012), (Sun et al., 2015), (Maillo et al., 2015). To illustrate the applicability of our method in reality, consider a theoretical computation for a data set consisting of 1 billion 100-dimensional samples using a brute-force nearest neighbor search on a modern high-end GPU like the NVIDIA A100:

- Distance Computation: Each distance calculation between two 100-dimensional points involves approximately 300 operations. Therefore, for 1 billion points, the total number of operations required is approximately 3×10^{11} operations.
- GPU Capability: The NVIDIA A100 GPU is capable of performing around 19.5 teraflops (FP32), meaning it can process approximately 19.5 x 10^{12} operations per second.
- Estimated Computation Time: Assuming full utilization of the GPU, the time required to compute the distances for 1 billion points would be: $\frac{3 \times 10^{11} \ operations}{19.5 \times 10^{12} \ operations/sec} \approx 15.3 \ ms$

This estimate assumes ideal conditions with full GPU utilization and does not account for additional factors such as memory transfer, indexing, or reduction operations required to identify the nearest neighbors. Even with some overhead, these numbers demonstrate that our method, leveraging the power of modern GPUs, is not computationally prohibitive. It is capable of handling large-scale data sets efficiently.

Our approach officially introduces three new hyper-parameters: parameter k, which controls the number of neighbors in knn model, the parameter $blob_{tr}$ which controls the close neighbors threshold and parameter $spread_{tr}$ which controls the target spread threshold for close neighbors. However, we treat $blob_{tr} = 75\%$ and $spread_{tr} = 0.5$ as constants and assess only parameter k. While we find that setting $k = 7$ typically provides good performance, there can be better choices for specific data sets. In general, higher k, e.g. 10, performs better for data sets with a higher number of features while smaller k, e.g. 5, tends to perform better for data sets with a smaller number of samples. The latter makes sense taking into account that higher k causes a stronger reduction of samples - which for data sets already small in size can be very detrimental. With this in mind, searching for optimal hyper-parameters values for a specific dataset can be reduced to variable k, with the optimal parameter being most likely in the range we performed the experiments ($k \in \{5, 10\}$).

As shown by Blagus and Lusa (2013), SMOTE - based techniques are expected to introduce bias and perform worse than RU for high-dimensional setting. On the other hand, random under-sampling does not asses the importance of samples that are removed and we consider that SUS and SUSiter have an edge here. Our experiments suggest that SMOGN does under-perform on synthetic imbalanced high-dimensional data compared to standard data sets.

On a higher level, there is a potential for applying SUS to contemporary forms of learning, such as federated learning, where data is distributed across multiple clients with decentralized data ownership and privacy concerns (Tang et al., 2022). In these scenarios, data imbalances at the client level can result in skewed global models that perform poorly on under-represented target ranges (Tang et al., 2024), (Tang et al., 2021). By integrating SUS into federated learning, it becomes possible to locally balance datasets before aggregating model updates, thereby enhancing the overall performance of the global model. Beyond federated learning, SUS can also be advantageous in distributed computing environments, and any context where maintaining a balanced representation of data is critical for the success of predictive modeling tasks.

We introduced both SUS and SUSiter algorithms, illustrating that they typically surpass existing techniques in performance. Detailed rationale behind this method is presented in Section 3. Specifically, we have incorporated theoretical insights from information theory, accompanied by entropy demonstrations, to further explain the underlying mechanisms of the method. Across standard data sets, SUS exhibits a slightly superior performance compared to SUSiter, whereas for synthetic high-dimensional datasets, SUSiter tends to outperform. Image data sets seem to be suitable for both algorithms. Generally, problems characterized by a higher number of features and samples prove more suitable for the effective application of both these algorithms.

6. Conclusion

In this manuscript, we introduce an innovative approach for selective under-sampling termed SUS, alongside its iterative counterpart SUSiter, designed to address the challenges associated with imbalanced regression. The SUS algorithm represents an under-sampling technique that takes into account the significance of each sample within a data set, considering both feature and target levels. The core concept revolves around the recognition that certain samples serve as more important representatives of the data set than others. SUSiter, the iterative version, makes use of all available data samples while maintaining the advantages of under-sampling. It achieves this by selecting a different but carefully chosen subset of data in each iteration of the learning process.

The key contributions of this paper are the proposal of a new pre-processing method SUS and its iterative version SUSiter.
We evaluate results using 15 standard, 5 synthetic high-dimensional data sets, specifically designed to represent different kinds of data distributions, and 2 complex age estimation image data sets, on neural networks as learners. Experiments on the evaluation data sets show that SUS/SUSiter typically outperform SMOGN, random under-sampling and performance on the original data set. The difference is more significant on data sets with high-feature numbers and high-dimensional data sets. SUS performs slightly better than SUSiter for standard data while SUSiter performs better with high-dimensional setting. Both SUS and SUSiter demonstrate superior performance with image data sets.

Future work involves the exploration of the following aspects: i) how SUS and SUSiter impact different learners (decision trees, support vector machines etc.) ?, ii) In comparison to SMOGN and other existing techniques, which types of problems are more apt for the application of SUS/SUSiter methods? and iii) do SUS and SUSiter exhibit comparable efficacy when employed alongside advanced neural network architectures such as transformers (Vaswani et al., 2023)?

Appendix A.

A.1 Parameters

Default values for SMOGN are used (Kunz, 2020): $k = 5$ specifies the number of neighbors to consider for interpolation in over-sampling, $pert = 0.02$ represents the amount of perturbation to apply to the introduction of Gaussian Noise, balanced sampling is selected, replacement is not selected in under-sampling and relevance function threshold is set to be 0.8, as in the original paper (Branco et al., 2017). Random under-sampling was done with a factor corresponding to the automatic reduction performed by SUS. Each data set's samples were selected by a different amount in SUS, and for a fair comparison, we took each data set's reduction factor in SUS to be used by random under-sampling technique.

A.2 Learning parameters

Architectures tested for standard and high-dimensional data sets: (32, 16, 8), (8, 4, 2), (20, 5), (10, 5). Listed architectures represent a number of hidden layers and the corresponding number of neurons per layer. Training is run for 300 epochs, we use Adam optimization (Kingma & Ba, 2014), and a learning rate of 10^{-2}.

A.3 IMDB-WIKI

The IMDB-WIKI dataset (Rothe et al., 2018) is a comprehensive collection of facial images designed for age estimation from a single image input. The original dataset includes a total of 523,000 face images, each annotated with corresponding age labels. Of these, 460,700 images were sourced from the IMDB website, with an additional 62,300 images obtained from Wikipedia. The dataset's creation follows the procedure outlined in (Yang et al., 2021), which begins with an initial curation step to filter out images with low face scores (Rothe et al., 2018). Following this, a balanced validation and test set is constructed across the supported age range.

After curation, the final dataset consists of 191,500 images for training, with 11,000 images each allocated to validation and testing. To ensure consistency, the dataset is divided into 1-year age bins, covering ages from 0 to 186 years.

For data pre-processing, the images are first resized to 224×224 pixels. During training, we apply a standard data augmentation strategy (He et al., 2016), which includes zero-padding each image with 16 pixels on all sides. The images are then randomly cropped back to their original size, followed by horizontal flipping. Finally, the images are normalized to a pixel value range of [0, 1].

A.4 AgeDB

The AgeDB dataset (Moschoglou et al., 2017) is a carefully curated, in-the-wild age database characterized by accurate, noise-free labels. Like the IMDB-WIKI dataset, its primary purpose is to estimate age based on visual appearance. The original dataset contains 16,488 images. Following a construction approach similar to that used for IMDB-WIKI, as described in (Yang et al., 2021), the AgeDB training set includes 12,208 images, covering an age range from 0 to 101 years. The dataset is organized into bins, with the most populated bin containing 353 images and the least populated bin containing just one image. The validation and test sets are carefully balanced, each containing 2,140 images.

As with IMDB-WIKI, the AgeDB images are resized to 224×224 pixels during pre-processing. The dataset also undergoes the same pre-processing steps used for IMDB-WIKI, ensuring consistency across both datasets.

A.5 ResNet

For all experiments involving the IMDB-WIKI and AgeDB datasets, we utilize the ResNet-50 model (He et al., 2016). Each model is trained over 90 epochs using the Adam optimizer (Kingma & Ba, 2014), starting with an initial learning rate of 10^{-3}, which is reduced by a factor of 0.1 at the 60th and 80th epochs. The batch size is consistently set to 256.

References

Blagus, R., & Lusa, L. (2013). Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, *14*, 106.

Branco, P., Ribeiro, R. P., & Torgo, L. (2016). Ubl: an r package for utility-based learning. *ArXiv*, *abs/1604.08079*.

Branco, P., Torgo, L., & Ribeiro, R. P. (2017). SMOGN: a pre-processing approach for imbalanced regression. In Luís Torgo, P. B., & Moniz, N. (Eds.), *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, Vol. 74 of *Proceedings of Machine Learning Research*, pp. 36–50. PMLR.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., & Lin, H. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877–1901. Curran Associates, Inc.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *ArXiv*, *abs/2005.12872*.

Castro, C. L., & Braga, A. P. (2013). Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(6), 888–899.

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.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277.

Fernández, A., Garca, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets* (1st edition). Springer Publishing Company, Incorporated.

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., & Olah, C. (2021). Multimodal neurons in artificial neural networks. *Distill*, *6*(3), e30.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE.

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., & Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, *596*, 583 – 589.

Kamalov, F. (2020). Kernel density estimation based sampling for imbalanced class distribution. *Inf. Sci.*, *512*, 1192–1201.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.. arxiv:1412.6980 Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

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

Kunz, N. (2020). SMOGN: Synthetic minority over-sampling technique for regression with gaussian noise..

Li, S., & Amenta, N. (2015). Brute-force k-nearest neighbors search on the gpu. In Amato, G., Connor, R., Falchi, F., & Gennaro, C. (Eds.), *Similarity Search and Applications*, pp. 259–270, Cham. Springer International Publishing.

Maillo, J., Triguero, I., & Herrera, F. (2015). A mapreduce-based k-nearest neighbor approach for big data classification. In *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 02*, TRUSTCOM-BIGDATASE-ISPA '15, p. 167–172, USA. IEEE Computer Society.

Mitchell, T. M. (1997). *Machine learning*, Vol. 1. McGraw-hill New York.

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.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, pp. 807–814.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ribeiro, R. P. (2011). *Utility-based Regression*. Ph.D. thesis, Portugal.

Rothe, R., Timofte, R., & Gool, L. V. (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.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379–423.

Steininger, M., Kobs, K., Davidson, P., Krause, A., & Hotho, A. (2021). Density-based weighting for imbalanced regression. *Machine Learning*, *110*, 2187–2211.

Sun, K., Kang, H., & Park, H.-H. (2015). Tagging and classifying facial images in cloud environments based on knn using mapreduce. *Optik*, *126*(21), 3227–3233.

Tang, Z., Hu, Z., Shi, S., ming Cheung, Y., Jin, Y., Ren, Z., & Chu, X. (2021). Data resampling for federated learning with non-iid labels..

Tang, Z., Zhang, Y., Shi, S., He, X., Han, B., & Chu, X. (2022). Virtual homogeneity learning: Defending against data heterogeneity in federated learning..

Tang, Z., Zhang, Y., Shi, S., Tian, X., Liu, T., Han, B., & Chu, X. (2024). Fedimpro: Measuring and improving client update in federated learning..

Torgo, L., & Ribeiro, R. (2007). Utility-based Regression. In JN, K., de Mántaras RL, K. J., Matwin, S., Mladenic, D., & Skowron, A. (Eds.), *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, Vol. 4702 of *Lecture Notes in Artificial Intelligence*, pp. 597–604. Springer.

Torgo, L., Ribeiro, R. P., Pfahringer, B., & Branco, P. (2013). Smote for regression. In Correia, L., Reis, L. P., & Cascalho, J. (Eds.), *Progress in Artificial Intelligence*, pp. 378–389, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vandal, T., Kodra, E., Ganguly, S., Michaelis, A. R., Nemani, R. R., & Ganguly, A. R. (2017). Deepsd: Generating high resolution climate change projections through single image super-resolution. *CoRR*, *abs/1703.03126*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need..

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J. (2016). Training deep neural networks on imbalanced data sets. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4368–4374.

Yang, Y., Zha, K., Chen, Y.-C., Wang, H., & Katabi, D. (2021). Delving into deep imbalanced regression. In *International Conference on Machine Learning (ICML)*.

Zhang, C., Li, F., & Jestes, J. (2012). Efficient parallel knn joins for large data in mapreduce. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, p. 38–49, New York, NY, USA. Association for Computing Machinery.

Zhang, S., & Karunamuni, R. J. (1998). On kernel density estimation near endpoints. *Journal of Statistical Planning and Inference*, *70*(2), 301–316.