Applied Soft Computing 115 (2022) 108288

Contents lists available at ScienceDirect

Applied Soft Computing

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

KNNOR: An oversampling technique for imbalanced datasets

Ashhadul Islam [a,*], Samir Brahim Belhaouari [a], Atiq Ur Rehman [a], Halima Bensmail [b]

[a] Division of Information and Computing Technology, Hamad Bin Khalifa University, Qatar
[b] Qatar Computing Research Institute, Qatar

ARTICLE INFO

Article history:
Received 9 May 2021
Received in revised form 29 November 2021
Accepted 6 December 2021
Available online 10 December 2021

Keywords:
Data augmentation
Machine learning
Imbalanced data
Nearest neighbor
Support Vector Machines

ABSTRACT

Predictive performance of Machine Learning (ML) models rely on the quality of data used for training the models. However, if the training data is not balanced among different classes, the performance of ML models deteriorate heavily. Several techniques have been proposed in the literature to add some semblance of balance to the data sets by adding artificial data points. Synthetic Minority Oversampling Technique(SMOTE) and Adaptive Synthetic Sampling(ADASYN) are some of the commonly used techniques to deal with class imbalance. However, these approaches are prone to 'within class imbalance' and 'small disjunct problem'. To overcome these problems, this article proposes an advanced algorithm by studying the compactness and location of the minority class relative to other classes. The proposed technique called K-Nearest Neighbor OveRsampling approach (KNNOR) performs a three step process to identify the critical and safe areas for augmentation and generate synthetic data points of the minority class. The relative density of the entire population is considered while generating artificial points. This enables the proposed KNNOR approach to oversample the minority class more reliably and at the same time stay resilient against noise. The proposed method is compared with the ten top performing contemporary oversamplers by testing the accuracy of classifiers trained on augmented data provided by each oversampler. The experimental results on several common imbalanced datasets show that our method ranks first more consistently than the other state-of-art oversamplers. The proposed method is easy to use and has been made open source as a python library.

Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.6530837.v1.

1. Introduction

Machine learning techniques can help in extraction of useful knowledge from large, complex, heterogeneous and hierarchical data. When it comes to classification, various methods like neural network, naïve bayes, k Nearest Neighbors(KNN), Support Vector Machines(SVM) and several others have achieved good performance [1–3]. As these state-of-the-art models emphasize more on classification accuracy than the imbalanced nature of the input data, it is possible for a classifier to achieve high classification accuracy and not predict a single minority class correctly in case of an imbalanced dataset. Assuming a dataset with 0.1% negative cases, a simple classifier which predicts all data points to be positive will score a classification accuracy of 99.9%. However, in this case all the negative cases remain undetected. Thus, when the input data is imbalanced the classification will retain a bias towards the majority class and the decision boundary line will be biased towards the minority class samples [4–7]. Often the minority observations are treated as noise and ignored in classification. In some cases, most of the test data samples are classified into the majority group. As a result, the classification accuracy of minority class tends to be much lower than that of the majority class.

A problem often encountered in imbalanced datasets is the small disjuncts problem [8]. This is the case where some of the minority data points show very unique exceptional characteristics. The classifiers consider these points as outliers. If we take these edge cases together, they may constitute a significant proportion of the minority population. When these points are disregarded by the classifiers, it reduces an already shrinking minority population to an even smaller count [9]. Another issue is the within class imbalance where subsets of the minority class may have much fewer examples than the other subsets of the same class. In the creation of any augmentation algorithm, it is imperative that algorithms are facilitated to tackle the small disjuncts and within class imbalance problems. In light of the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: aislam@hbku.edu.qa (A. Islam).

https://doi.org/10.1016/j.asoc.2021.108288

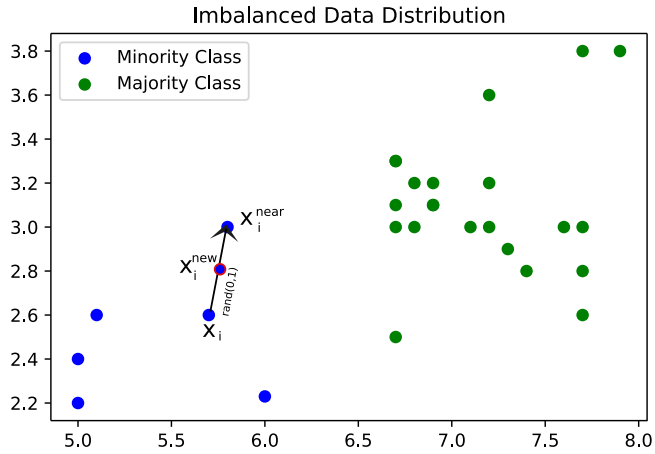

Fig. 1. SMOTE — the fundamental augmentation algorithm. x_i^{new} is the generated point at a random distance between x_i and x_i^{near}.

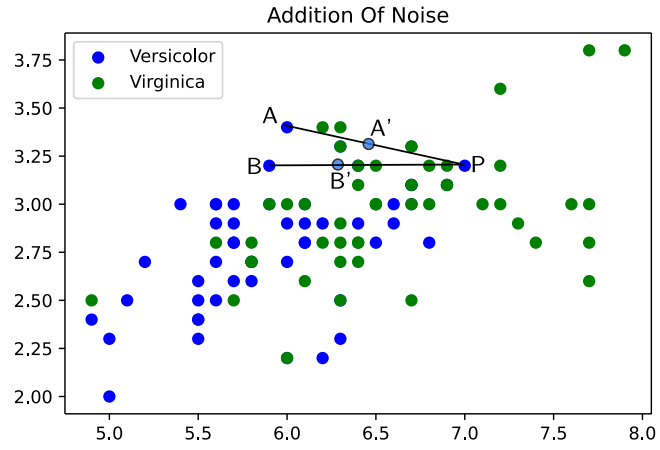

Fig. 2. Addition of Noise due to SMOTE. Point P is the source minority point while A and B are neighboring minority points. As P is in a region populated by majority points, the artificial points generated by joining A and P(A') and by joining B and P(B') tend to add more noise to the data.

aforementioned shortcomings, this research focuses on the data distribution problem and presents a novel data over sampling approach, the K-Nearest Neighbor OveRsampling Approach(KNNOR) for imbalanced datasets which consists of the following three steps:

1. The data points belonging to the minority class are sorted based on the distance to their *k*th nearest neighbor. This enables the algorithm to reach the denser minority regions and ignore outliers or noisy data.
2. In the second step, the sorted list of points and their k nearest neighbors (of the same class) are used to create an artificial data point. The procedure starts with an origin point and finds another point at a random distance between the origin and its first nearest neighbor. The new point is next considered, and another point is found at a random distance between the new point and the origin's second closest neighbor. After k such iterations, the point formed is deemed an artificially generated candidate.
3. In this step the artificially generated candidate from step 2 is tested with a k-nearest neighbor classification algorithm to check if it belongs to the class that is being oversampled. If it passes the test, the data point is retained, otherwise it is rejected.

The rest of this paper is organized as follows. Section 2 presents related work on augmentation on imbalanced datasets. Section 3 elaborates the framework of the method proposed. Section 4 presents the experiment design while Section 5 enumerates the results to substantiate the effectiveness of our proposed method. The conclusions and future work for this research are summarized in Sections 6 and 7.

2. Related work

In the past decades, three popular approaches have been used to handle imbalanced data. The first approach is data-driven [10]- these are sampling based methods that change the initial distribution of minority and majority class in the training set, resulting a balanced distribution of each class. This approach will be discussed in detail later. The second approach is algorithm-based techniques. Some of the popular techniques are Tomek Link or Tlink [11], One Sided Selection (OSS) [12], Neighborhood cleaning Rule (NCL) [13], Bootstrap based Oversampling (BootOS) [14] to name a few. The third approach consists of hybrid methods [15]. Algorithm techniques are often bound to specific classifiers whereas data-level methods are more versatile since they can be universally applied [8]. Because the proposed method deals with data, the existing data-driven approaches are discussed in greater depth, as shown below.

Over the past few years of research different data driven methods have been designed to counter the class imbalance problem. Methods like oversampling and under sampling [16,17] modify the data to reduce the impact of class imbalance. Under sampling process randomly removes a fraction of the majority class samples to level the data distribution at the cost of information loss. Algorithms like BalanceCascade and EasyEnsemble [17] employ this technique. The over sampling process randomly duplicates minority data points in order to increase its count. This may often lead to overfitting. A more advanced over sampling technique is the Synthetic Minority Oversampling Technique (SMOTE) [18] that creates new minority class samples to balance the data distribution. The SMOTE algorithm generates a new data point by selecting a point on a line connecting a randomly chosen minority class sample and one of its k nearest neighbors.

If x_{new} is the synthetic point generated, it can be expressed as

$$x_i^{new} = x_i + (x_i - x_i^{near}) * \alpha_i \quad (1)$$

where α_i is independent and identically distributed number uniformly distributed on [0, 1], the point x_i is a data point belonging to the minority class and x_i^{near} is one of the k nearest neighbors of x_i, by considering only minority class, selected at random. The random number α_i is multiplied with each element of $(x_i - x_i^{near})$ which effectively results in choosing a point on the line joining x_i and x_i^{near} at a distance of α_i (see Fig. 1).

The algorithm harbors some weaknesses when it comes to dealing with noise and imbalance. Fig. 2 shows the distribution of sepal length and sepal width of the Versicolor and Virginica species of the Iris dataset [19]. A and B are two points belonging to the Versicolor class while P is also another point pertaining to the same class, located in a region that is predominantly made up of Virginica data points. SMOTE focuses on creating synthetic data points while neglecting the presence of noise around P. As a result, while generating artificial points with reference to points A and B, the algorithm may randomly select P as one of their nearest neighbors and generates artificial points (A' and B') in Virginica zone, thereby adding to the noise in the dataset. This happens because SMOTE chooses a target datapoint with uniform probability, resulting in selecting noisy data points in the vicinity

of the other class instances and consequently adding more points around the other class instances. Since SMOTE does not distinguish overlapping class regions from so called safe zones, it is prone to amplifying the noise already present in the data [20–23]

Despite its deficiencies, SMOTE is simple and easy-to-use to generate artificial points. It is widely used by researchers and practitioners and has motivated many variants. The work of Kovacs [24] is significant as it compares and evaluates 85 different methods of minority oversampling, using 104 imbalanced datasets. Based on the findings, Kovacs [24] lists the top 10 oversamplers that we employed in our study to compare to our proposed algorithm (KNNOR). These oversampling algorithms are polynom-fit-SMOTE [25], Proximity Weighted Synthetic (ProWSyn) oversampling [26], SMOTE-Iterative-Partitioning Filter (SMOTE-IPF) [22], Lee [27], Synthetic Minority Oversampling Based on samples Density (SMOBD) [28], Geometric SMOTE (G-SMOTE) [29], CCR [30], Learning Vector Quantization (LVQ)-SMOTE [31], Assembled SMOTE [32] and SMOTE — Tomeklinks [33]. A brief overview of each approach is presented before discussing our approach.

The polynom-fit-SMOTE [25] method uses polynomial fitting methods to generate artificial data. It proposes 4 new approaches including star topology, bus topology, polynomial curve topology and mesh topology. ProWSyn [26] follows the idea of allocating weights to each minority sample according to its importance. It generates effective weight values for each minority data point by considering its distance from the class boundary. SMOTE-IPF [22] tries to deal with noisy and borderline samples in the minority dataset by providing an iterative ensemble-based noise filter. Lee [27] proposes an over-sampling technique with rejection, where he checks whether the generated artificial data point is surrounded by a majority of data points from the minority class after applying SMOTE. SMOBD [28] proposes an algorithm that considers the density and distribution information about the dataset before creating synthetic minority samples. In G-SMOTE [29] each new artificial sample is generated within the k-nearest neighbor locality of the minority data point under consideration. CCR [30] removes majority data points from the neighborhood of minority samples and then generates artificial data around minority data points that are closer to the majority ones. Learning Vector Quantization(LVQ)-SMOTE [31] generates synthetic samples using codebooks obtained by LVQ, which is a supervised classification algorithm. Assembled SMOTE [32] creates local linear partitions in and around the data as a processing step, then applies SMOTE on separate clusters of the data separated by these partitions. SMOTE — Tomeklinks [33] first applies SMOTE to synthesize artificial data and then removes pairs of majority–minority data points if they are identified as noise or borderline data.

Most of the previously described algorithms uses SMOTE at the preprocessing level or at the core of the algorithm. Therefore, tackling the limitations of SMOTE is useful and necessary. As we stated before, the aim of this paper is to address the small disjunct and within class imbalance limitations. The next section will describe our approach for solving the two problems.

Our contribution is summarized as the following:

- Introduction of a method to identify the points that better represent the minority dataset. (Section 3.1)
- Usage of multiple minority data points while generating the new data sample to circumvent the problem of small disjunct. (Section 3.2)

Table 1
Sample distance of each point to its nearest neighbor.

Origin	1st Closest	Distance	2nd Closest	Distance	3rd Closest	Distance
x_1	x_3	1	x_4	7	x_2	12
x_2	x_4	5	x_3	9	x_1	12
x_3	x_1	1	x_5	2	x_2	9
x_4	x_5	3	x_2	5	x_1	7
x_5	x_3	2	x_4	3	x_1	15

3. Framework

Before detangling the algorithm, we define an imbalanced dataset as a set which has a high number of data belonging to a specific class and a comparatively smaller representation for the other class. Even if the majority class is two times the size of the minority class, it is still considered to be an imbalanced dataset. In the remaining of this section, we will explain the proposed algorithm in detail. The algorithm is divided into three parts: (1) Pre-Augmentation, (2) Augmentation and (3) Validation.

3.1. Pre-augmentation step

This part of the algorithm focuses on finding a subset of the minority class datapoints. Points in this subset are closer to their kth nearest neighbor belonging to the same class. The value of k and population size of the subset(d) varies across datasets. This is elaborated in Section 3.1.1. The whole dataset can thus be reduced to a single array of values. Each of these values are the distance of each data point to its kth nearest neighbor. Consider a case of data in $\mathbf{R}^n$. The n dimensional data can be transformed to a single scalar representation to measure the compactness using a distance measure like 'Euclidian distance', which computes the distance of each observation in $\mathbf{R}^n$ to its kth closest neighbor, named d_k. The concept of transformation can be summarized as the following function:

$$d_k : \mathbf{R}^n \rightarrow \mathbf{R} \tag{2}$$

Where $d_k \in \mathbf{R}$, is a scalar containing the distance of each observation in $\mathbf{R}^n$ to its kth closest neighbor. During augmentation, the KNNOR algorithm applies a threshold to ensure the compactness of data.

Let us understand this step with the help of an example of a minority dataset containing 5 points $[x_1, x_2, x_3, x_4, x_5]$. Table 1 shows the distance of each point to its 3 nearest neighbors. The first column shows the distance to the 1st closest neighbor for each of the datapoints. Consider the case where the value of k is 3. This implies that the dataset can be represented by each points distance to its 3rd nearest neighbor which is the third column of Table 1. As can be seen from the same, the distance of x_1 to its 3rd nearest neighbor is 12, the distance of x_2 to its 3rd nearest neighbor is also 12 and the distance of x_3 to its 3rd nearest neighbor is 9 and so on. The points $[x_1, x_2, x_3, x_4, x_5]$ can now be ordered according to this distance. x_4 which is at a distance of 7 from its 3rd nearest neighbor, would top the list, followed by $x_3, x_2, x_1 and x_5$. Thus the sorted list of points based on their distance to the third nearest neighbor would be $[x_4, x_3, x_2, x_1, x_5]$. In order to complete the filtration, a subset of these points are taken and provided to the next steps of the algorithm. For example, if the proportion is 80% then the top 4 points $[x_4, x_3, x_2, x_1]$ will be passed to the next steps of the algorithm. Similarly, if the proportion is 60%, then the points used to generate the artificial points would be
$[x_4, x_3, x_2]$

With varying values of k and d, the order as well as count of points considered, change. For example, if k is 2 then the list becomes the sequence of points:

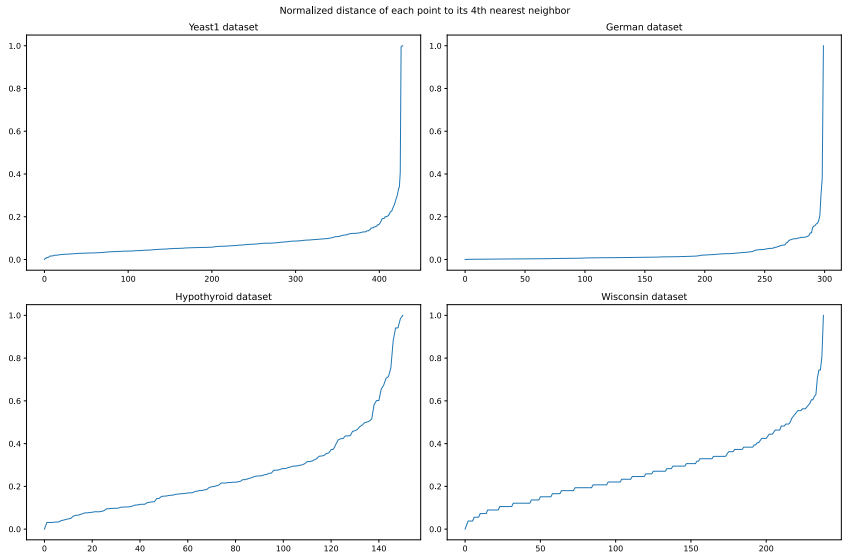

Fig. 3. Distance distribution of each point from its 4th nearest neighbor for Yeast1, German, Hypothyroid and Wisconsin datasets [19]. X axis represents the points while the Y-axis illustrates the distance of each point to its 4th nearest neighbor.

$[x_3, x_5, x_4, x_1, x_2]$

and if the distance threshold(d) is 80%, the top 80% of the sorted list is used:

$[x_3, x_5, x_4, x_1]$.

In this way the algorithm filters out noise and outliers, concentrating on densely populated minority zones. The degree of concentration is configurable and can be calculated as explained in 3.1.1. This algorithm helps to identify points using which generation of artificial data points would be more effective. This capability is further demonstrated in the following subsections.

Algorithm 1 KNNOR Filtering

Input Training dataset S;
Number of nearest neighbors k;
Output List of points to be used for augmentation, ($Sorted_{min}$), List of majority points S_{maj} and List of minority points S_{min};
Divide the training dataset S into S_{maj} containing features of the majority population and
S_{min} containing features of the minority population
for every data point p in S_{min} do
  Calculate distance of p to its k_{th} nearest neighbor
  Append the distance to a list L_{dist}
Sort the indices of list L_{dist} in descending order L_{sorted}
Get the value of the proportion of minority population to be used(d) using Algorithm 1a (3.1.1)
Return the top d% of the sorted list ($Sorted_{min}$), S_{maj}, S_{min};

3.1.1. Distance threshold calculation

The value of the distance threshold(d) is an important consideration and needs to be calculated judiciously for each dataset. Fig. 3 shows the distance of each point to its 4th nearest neighbor for Yeast1, German, Hypothyroid and Wisconsin datasets [19]. The distance distribution is different for each dataset and a method needs to be devised to find the threshold point at which the distance starts increasing rapidly. In order to calculate this threshold point, the distribution is first smoothed using the Savitzky–Golay filter [34]. In this operation the data is treated with a set of m convolution co-efficients C_i as shown in the expression below

$$Y_j = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i y_{j+i}, \quad \frac{m+1}{2} \leq j \leq n - \frac{m-1}{2} \tag{3}$$

where the data consists of a set of points $x_j, y_j, j = 1, \ldots, n$, and x_j is an independent variable while y_j is an observed value. The effect is seen in Fig. 4. After the distribution is smoothed, the target is to find the point from where the distance rapidly increases. In order to calculate this, the sin of tan inverse of slope of every point is calculated.

$$\sin\left(\tan^{-1}\left(\frac{dy}{dx}\right)\right) \tag{4}$$

The results are shown in Fig. 5. The maxima of this distribution is the threshold point after which all points must be ignored. Using the location of the maxima, the threshold point is highlighted in Fig. 6. The green points are close to their 4th nearest neighbor and can be used to create artificial data points. The highlighted point in blue is the threshold after which the points are farther away from their 4th nearest neighbor and should not be used. The process is explained in Algorithm 1a. Some more scenarios of different values of distance threshold(d) and number of neighbors(k) are given in Section 3.1.2.

Algorithm 1a Getting the proportion(d) of minority datapoints to be used

Input Number of nearest neighbors k;
Sorted list L_{sorted} of distance to k_{th} nearest neighbor;
Output Value of d;
$L_{smoothed}$ = Savitzky–Golay filter on L_{sorted} to smooth values
dl = first differential of $L_{smoothed}$
$sin\_values = sin(tan^{-1}dl)$
max_pos = position of maxima in sin_values
$d = \frac{max\_pos}{Length(dl)}$
Return d

3.1.2. Handling outliers

It is important to note that the choices of k (number of neighbors) and d (percentage of points to be considered) play an important role in dealing with noise and outliers. Here we use the Versicolor and Virginica data from the iris dataset [19] to demonstrate how sensitive the filtering technique is to noise. In the first case, the value of d is kept constant while k has changed. The result is seen in Fig. 7 where the value of k has changed to 1, 2, 4, 6 and 8. As the value of d is kept fixed at 0.7, the

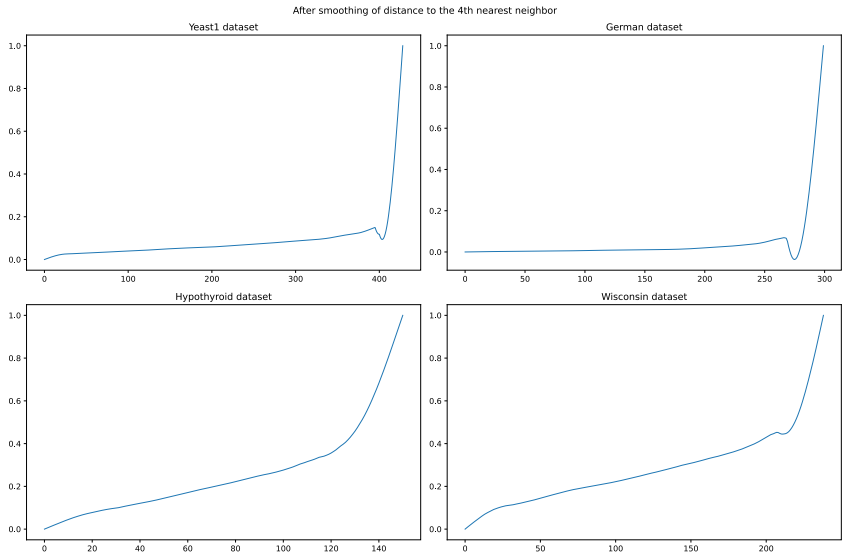

Fig. 4. Smoothed distance distribution of each point from its 4th nearest neighbor for Yeast1, German, Hypothyroid and Wisconsin datasets [19] using the Savitzky–Golay filter. Window size is 61 and polynomial order is 3.

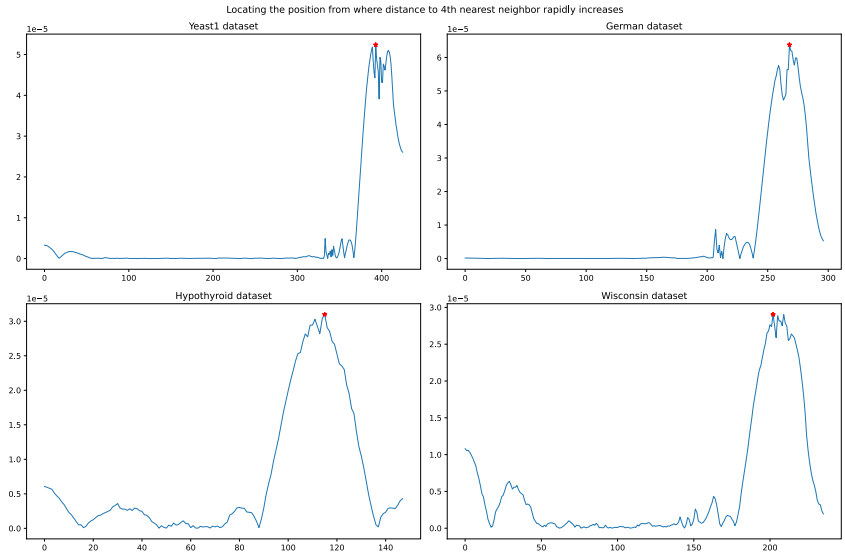

Fig. 5. The plot of $\sin\left(\tan^{-1}\left(\frac{dy}{dx}\right)\right)$ for each point in the distribution. The location of the minima is considered to be a threshold from where the distance to the 4th nearest neighbor abruptly increases.

number of points eliminated is the same (14). It is interesting to note the position of the remaining 36 points for different values of k. For comparison purpose, we can see that k = 2, $d = 0.7$ in Fig. 7c does the best job at filtering outliers. In fact, when the filter distribution is used with k = 2, $d = 0.7$, then the two outliers marked in red with a black outline are ignored (Fig. 8). In all the other cases, as can be seen in Fig. 7 one or both of the two outliers are retained. Fig. 9 shows the corresponding case when the value of k is fixed and value of d is changing. In this case, the number of points remaining after filtration changes with the value of d. When d is small, the filtering algorithm only chooses the most centrally located points while with increasing d, the filtered data is spread more and more, covering a greater area of the data region. It is to be noted that the value of k and d are passed as parameters to the augmentation algorithm and can be set according to the spread of the data. The following section also tests the validity of the filtering algorithm on different types of noises.

3.1.3. Handling mislabeled classes

Mislabeled data are data instances incorrectly classified during the data generation process [35]. To illustrate the effectiveness of our filtering algorithm, we take the iris dataset [19] and random datapoints of the Versicolor class are marked as belonging to the Virginica category (Fig. 10). Those points are marked in pink (Fig. 10b). The ability of the filtering algorithm to overlook these mislabeled datapoints determine its success. Fig. 10c shows the datapoints selected when k is 4 and d is 0.5. In this scenario, 3 of the 5 misclassified points are included by the filtering algorithm. On the other hand, when the value of k is 2 and d is 0.2, all the mislabeled datapoints are correctly discarded by the algorithm.

3.1.4. Illustration on Haberman data

Fig. 11 shows the selection of minority points according to the distance threshold specifically for Haberman [19] dataset. Fig. 11a represents the imbalanced Haberman [19] dataset. Green squares represent majority, orange triangles represent the minority data points. Fig. 11b shows only the minority points represented by

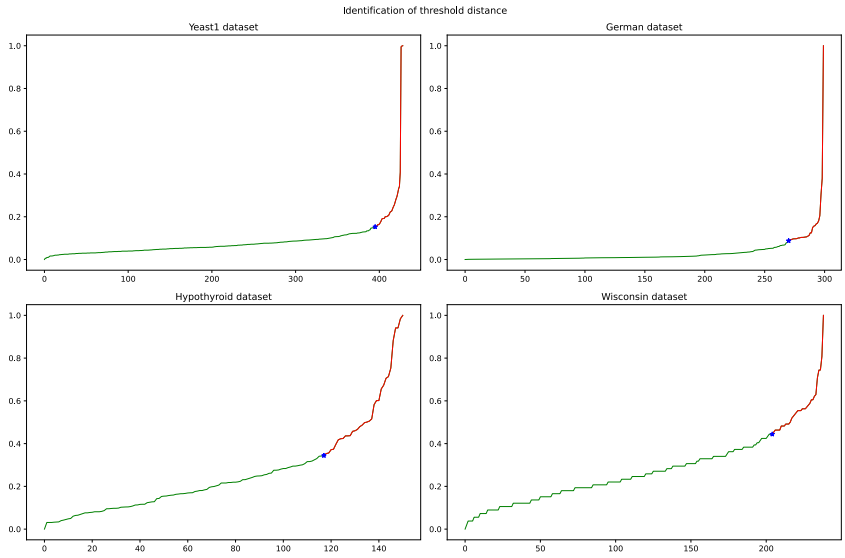

Fig. 6. Applying the threshold on original data. Points in green can be utilized to create the new data points. The points in red should not be used for augmentation as they are considerably distant from their fourth nearest neighbor. The blue star highlights the threshold.

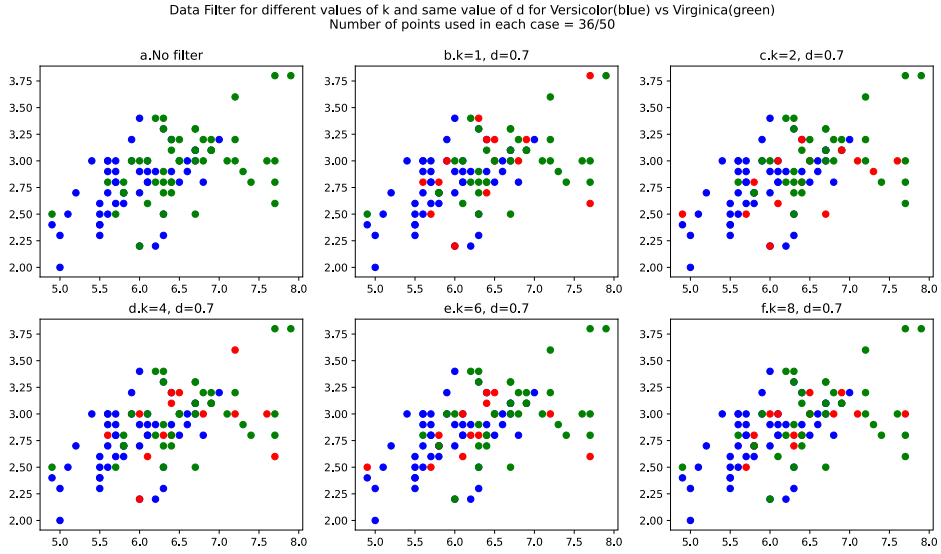

Fig. 7. The selection of datapoints from the Virginica class [19] for different values of k while keeping d fixed. Green points refer to the datapoints belonging to the Virginica class that will be used in augmentation step. Red points refer to the datapoints belonging to the same class those will not be used in the consequent augmentation steps.

orange triangles. Selection is made from these points on the basis of their distances from their kth nearest neighbor (3 in this case). In Fig. 11c, the threshold for selection is kept at 0.2, as a result only the top 20% points based on their distance to their 3rd nearest neighbor is selected. The points represented by the green triangles in Fig. 11c will be used in the consequent steps to generate artificial points while the red points will be ignored. Fig. 11d shows the selection of points when the threshold is changed to 0.8. As a result, 80% of the points (green triangles) are selected to be used for producing artificial data points.

3.2. Augmentation step

In the previous step, a set of points belonging to the minority class is generated. The points are chosen based on a dynamic threshold such that the points closest to their kth nearest neighbor get higher priority. This step, called the augmentation step is an iterative process where each point belonging to the above set is used as an origin. Along with the origin point, its k-nearest neighbors are also taken into account while generating the artificial point. A random value (α) between 0 and 1 is pre-fixed. This value is discussed in the subsection below. An artificial point is generated between the origin and its nearest neighbor at a distance of a random value between 0 and α. This artificially generated point is used to perform further computations. The 2nd artificial point is generated at a distance $random(0, \alpha)$ between the 1st artificial point and the 2nd closest neighbor to the origin point. As shown in Eq. (5), we continue generating artificial points for each nearest neighbor of the first origin point until all the k nearest neighbors are covered.

$\forall i \in [1, 2...k]$

$$x_{i+1}^{new} = x_i^{new} + (p_i - x_i^{new}) * \alpha_i \tag{5}$$

x_0^{new} = any safe point in the minority class

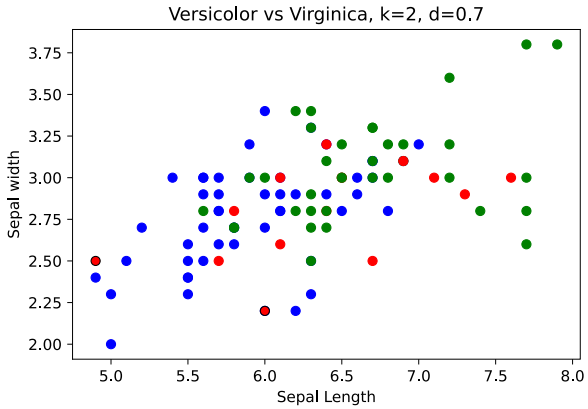

Fig. 8. The selection of datapoints from the Virginica class for k = 2 and $d = 0.7$. Green points refer to the datapoints belonging to the Virginica class that will be used in augmentation step. Red points refer to the datapoints belonging to the same class those will not be used in the consequent augmentation steps. Two red points can be seen with black outline. These are outliers that have been successfully filtered by the algorithm.

$p_i = i^{th}$ nearest neighbor of x_0^{new}

α_i are independent uniform random variable over $[0, M], M <= 1$

At each subsequent step, the artificial point generated at the previous step becomes the source point and a new point is generated on the straight line joining the source point and the kth nearest neighbor of the origin point at a distance $random(0, \alpha)$ from the source. The artificial point generated in the last iteration is considered to be the synthetic point generated out of this step. Fig. 12 shows a simulated imbalanced dataset and Fig. 13 shows the process of creation of a new data point. The point p_0 is chosen as the origin to generate the artificial datapoints. The value of k is set to 3 and the process is repeated for the 3 nearest neighbors of p_0. At first p_0 and p_1 are used to generate a synthetic point m_0.

$$m_0 = p_0 + (p_0 - p_1) * \alpha_0 \tag{6}$$

The artificially generated point m_0 is now used as the source point to calculate the next artificial point m_1 using the second closest neighbor of the origin point p_0.

$$m_1 = m_0 + (m_0 - p_2) * \alpha_1 \tag{7}$$

As can be seen in Fig. 6, after the 3rd and final step of augmentation

$$m_2 = m_1 + (m_1 - p_3) * \alpha_2 \tag{8}$$

The point m_2 is used as the final artificial points. The process is formulated in Algorithm 2 and further clarified in the flowchart at Fig. 17.

3.2.1. Usage of multiple artificial points as origin

The subsequent use of artificial points as the source to create the final artificial point allows the algorithm to penetrate deeper into the data space and alleviate the small disjunct problem. This is the case where specific subset of one class has much fewer samples than another subset of the same class. Fig. 14 shows the distribution of datapoints for an artificial dataset where the problem of small disjunct is apparent. There is a small cluster of minority data points and a larger one at certain distance. Randomly choosing a point in between an origin point belonging to the minority class and one of its neighbors of the same class increases the chances of creating more such small disjuncts. However, consecutive consideration of artificial data points with minority neighbors of the original data point ensures that the new points add to the existing clusters and not create new ones.

Algorithm 2 KNNOR Augmentation

Input Training dataset S_{maj}, S_{min}, Top d% of the sorted list($Sorted_{min}$);
number of neighbors k;
count of datapoints to be augmented aug_num;
Output Artificial point
Get the value of α using Algorithm 2a (3.2.2)
While aug_num>0 do
  For each point in ($Sorted_{min}$) do
    source = point
    For k nearest neighbor n of point do
      newpoint = point generated at a distance of $random(0, \alpha)$ on a line joining source and n
      source = newpoint
    End For
    if newpoint is valid(Algorithm 3)
      aug_num = aug_num-1
      Add newpoint to augmented_feature_list
      if aug_num == 0
        return augmented_feature_list
  End For
End While
Return augmented_feature_list

3.2.2. Value of α

The value of α or the distance at which the new point is augmented is a uniform random number chosen between 0 and an upper limit. This upper limit is based on a statistical study of the dataset.

In Fig. 15, which is a representation of sepal length and sepal width of two species of flowers, Setosa and Versicolor [19], the classes are distinct in their distribution but have a little overlap. This can be characterized by calculating the minimum intra class distance and the minimum inter class distance between the two. The distance is calculated by the Frobenius norm [36] given by

$$\|A\|_F = \left[\sum_{i,j} abs\left(a_{i,j}\right)^2\right]^{1/2} \tag{9}$$

Where A is the matrix of points and $a_{i,j}$ are the elements of the matrix. The Frobenius norm is generalized by the L^p space equation

$$\|x\|_p = \left(\sum_{i \in I} |x_i|^p\right)^{1/p} \tag{10}$$

In Fig. 15, the minimum distance between any two points in the Setosa class is 0.09 while the minimum distance between a point in the Setosa class and a point in the Versicolor class is 0.36. This means that the upper limit of α can be set to 0.36. Conversely, Fig. 16 shows the case where there is a lot of overlap. In this case, the value of α needs to be low in order to avoid creating noise. Fig. 16 shows the distribution between datapoints of Versicolor and Virginica class. The minimum distance between any two points in Versicolor is 0.0999 while the minimum distance between any point in the Versicolor class and Virginica class is also 0.0999. In this case, the maximum value of α needs to be upper bounded by 0.0999 to avoid addition of noise.

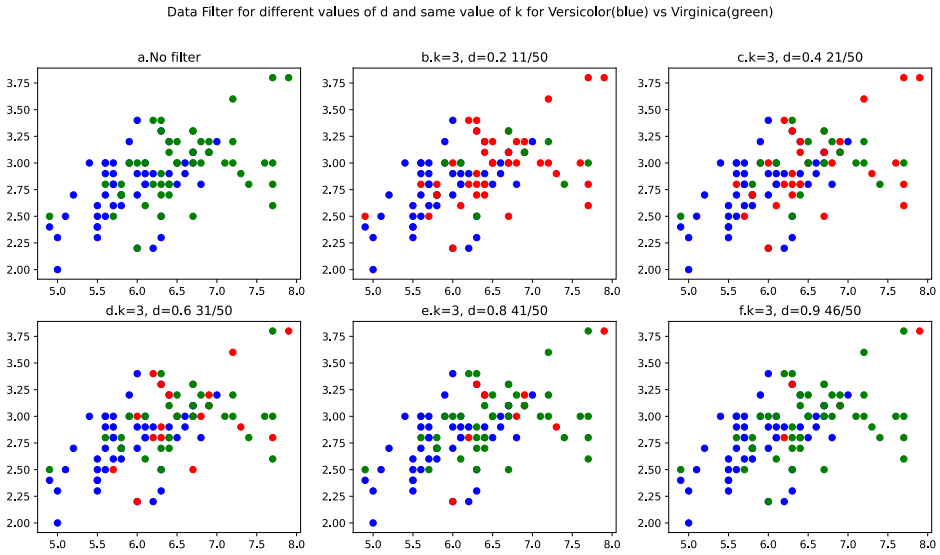

Fig. 9. The selection of datapoints from the Virginica class for different values of d while keeping k fixed. Green points refer to the datapoints belonging to the Virginica class that will be used in augmentation step. Red points refer to the datapoints belonging to the same class those will not be used in the consequent augmentation steps.

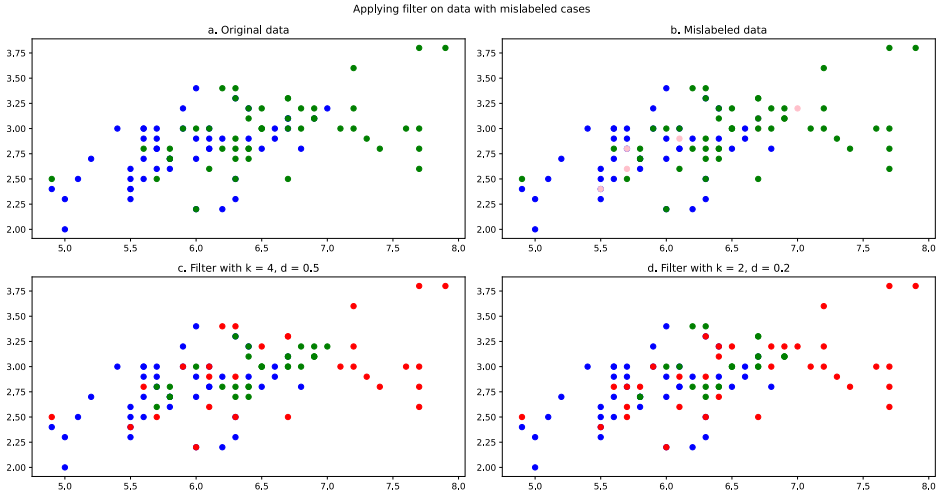

Fig. 10. The selection of datapoints from the Virginica class for different values of k and d after addition of noise due to mislabeling. Green points refer to the datapoints belonging to the Virginica class that will be used in augmentation step. Red points refer to the datapoints belonging to the same class and will not be used in the consequent augmentation steps. Pink points in Figure b represent the noise added by misclassifying versicolor datapoints as virginica.

Algorithm 2a Calculation Of α

Input Training dataset S_{maj}, S_{min};
Output Upper limit of α
all_dist = distance between points from S_{maj} to points in S_{min}
min_dist = minimum(all_dist)
return min_dist

3.3. Validation step

Once the augmented point is created it is validated using a k-nearest neighbor check. The k closest points to the newly created point are obtained and it is checked if all the k nearest neighbors belong to the same class as the artificial point. If any point is found to belong to a different class, the synthetic point is discarded.

Algorithm 3 KNNOR Validation

Input Augmented data point m;
k nearest neighbors of m;
Output True/False
can_use = True
For each neighbor of m do
  If neighbor belongs to same class as m
    can_use = can_use && True
  else
    can_use = False
    break
Return can_use

The entire procedure can be expressed through the flowchart in Fig. 18. It comprises of the data filtering step where points with neighbors at closer proximity are identified, the augmentation and the validation step.

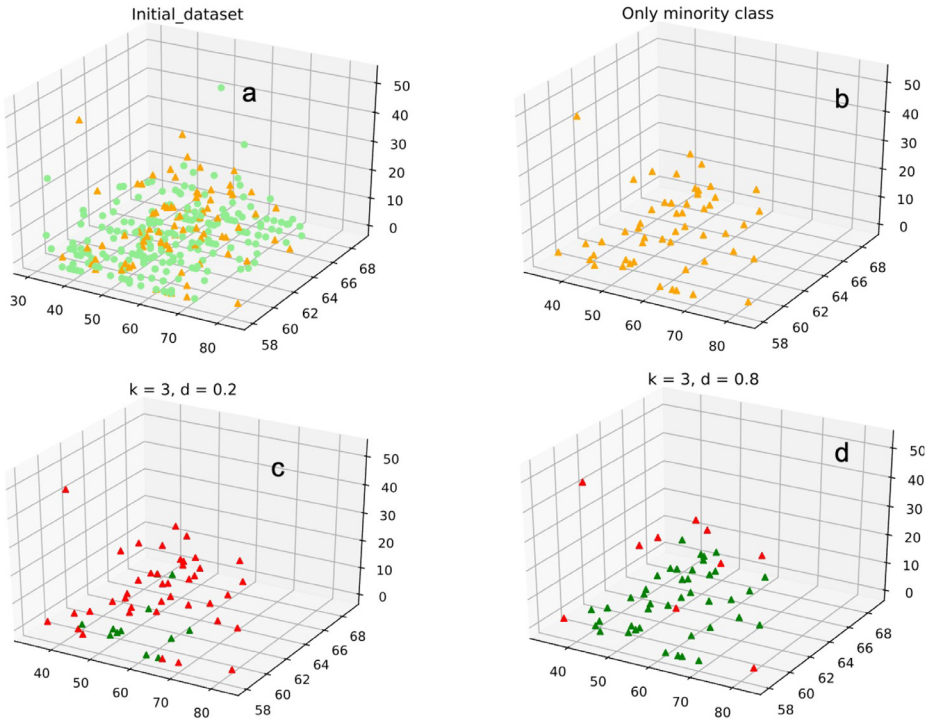

Fig. 11. Selection of minority data points based on threshold.

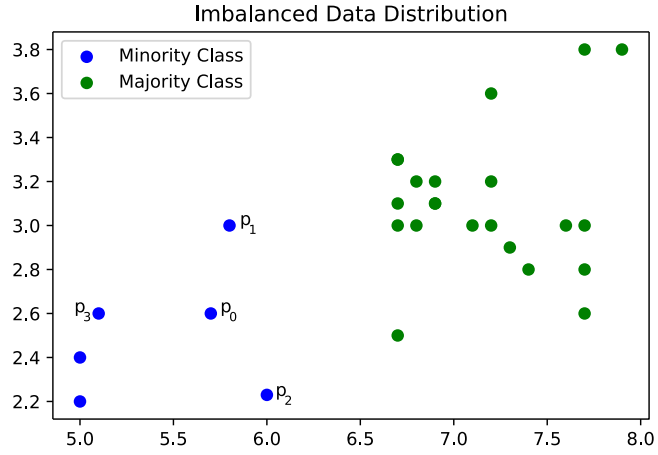

Fig. 12. Simulated dataset — before augmentation. p_0 is the source point from where augmentation will start. p_1, p_2 and p_3 are its 3 nearest neighbors in increasing order of distance.

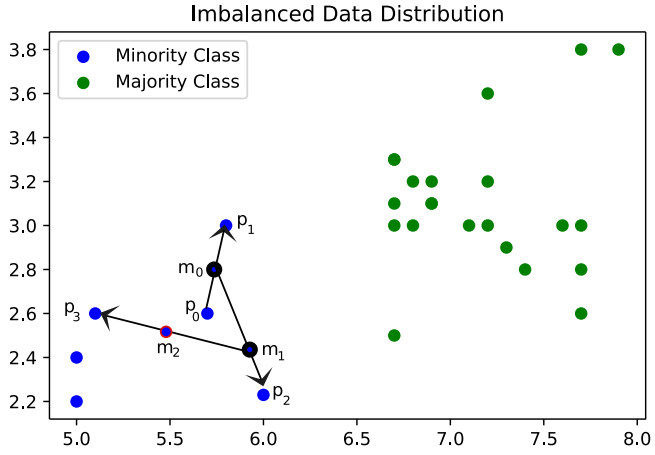

Fig. 13. The iterative process of creation of an augmented point m_2 with the help of three neighbors (p_1, p_2 and p_3), starting with p_0.

It is to be noted that the first step (Section 3.1) is executed till completion. The output of this step is a set of points. This set of filtered points is repeatedly used (Steps 2 and 3) until the required number of artificial datapoints is generated. In case the list of filtered data points is exhausted and the required number of artificial points have not been generated, the algorithm will again start from the top of the filtered list of points and generate new points using each of them. Since the points are generated at a distance of $random(0, \alpha_i)$ the points will be different at different iterations. At any point of iteration, if the required number of artificial points are created, the algorithm immediately breaks and returns the list of augmented points.

Fig. 19 gives a view of the different datapoints created based on the different values set for distance threshold. Fig. 19a shows the original Haberman dataset [19]. Fig. 19b shows the augmented points created for a distance threshold of 0.3. This implies that only the top 30% of data points that are nearest to their 3rd nearest neighbors are used for oversampling. As a result, the artificial points are also created at focused regions.

Fig. 19c is for a distance threshold of 0.5 and 19d is for 0.9. As can be seen, the distribution of augmented points is far more expansive for a distance threshold of 0.9 than for 0.3. This allows us to tune the degree of spread that we want to allow for the artificial points. The efficacy of KNNOR is further elaborated in a comparison with the top 4 oversamplers from Kovacs [37]. Fig. 20 shows the result of oversampling on the new thyroid1 data [19] using the polynom-fit-SMOTE [25] (Fig. 20a), ProWSyn [26] (Fig. 20b), SMOTE-IPF [22] (Fig. 20c), Lee [27] (Fig. 20d) and KNNOR(Fig. 20e & f). In case of KNNOR the distribution of the artificial points is done in a more controlled manner. The spread is kept tight and varies with the value of k as can be seen in Fig. 20e and Fig. 20f. Fig. 20e is the case where the

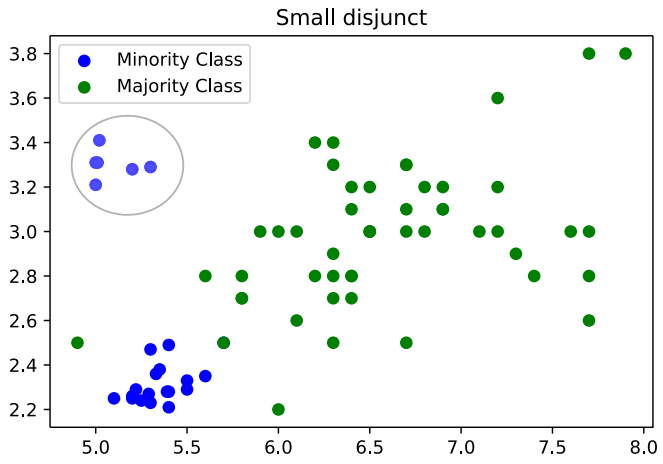

Fig. 14. Simulated imbalanced dataset — consisting of two clusters of minority datapoints. Highlighted cluster causes the small disjunct problem.

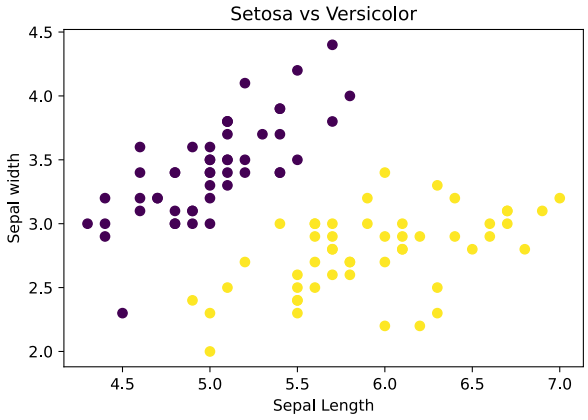

Fig. 15. The distribution of datapoints for Setosa (purple) and Versicolor (yellow) in the IRIS dataset [19]. Features used are sepal length and sepal width. Minimum distance between any two points in the Setosa class is 0.09 while the minimum distance between a point in the Setosa class and a point in the Versicolor class is 0.36.

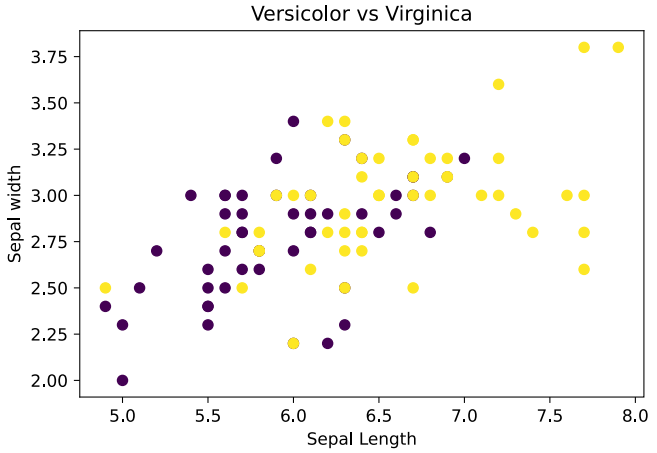

Fig. 16. The distribution of datapoints for Versicolor (purple) and Virginica (yellow) in the IRIS dataset [19]. Features used are sepal length and sepal width. Minimum distance between any two points in Versicolor is 0.0999 while the minimum distance between any point in the Versicolor class and Virginica class is also 0.0999.

number of neighbors used to generate new points is fixed at 7, as a result the final point is generated after 6 iterations and hence is

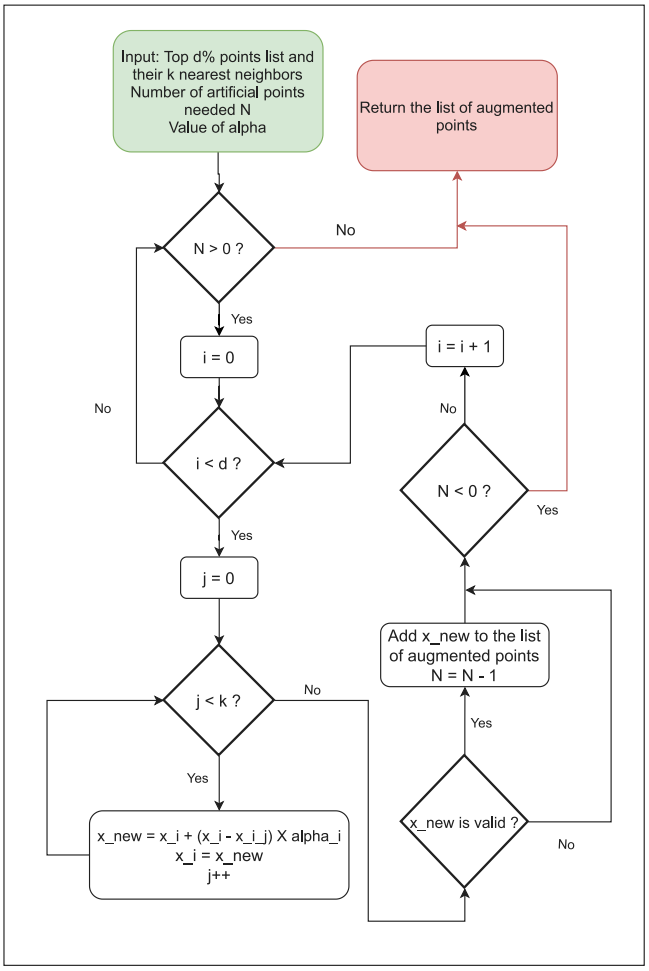

Fig. 17. Flowchart showing the operation of the Augmentation step. x_new is the artificial point generated at each step. x_i_j is the *j*th nearest neighbor to x_i.

a little spread out. Fig. 20f on the other hand displays the result of keeping the value of k as 2, resulting the new points being closer to the original minority points. In both cases though, the distance threshold is kept at 0.9 which means that 90% of the minority points closest to their *k*th neighbor are used. This plays a key role in balancing the augmentation process. In cases where the value of k is low, a higher value of distance threshold ensures that a greater number of minority points are covered. This is further illustrated in Fig. 21 which shows the generation of new points using only KNNOR on the same dataset. The two parameters that have been tweaked are the number of neighbors and the vaue of dist_thres (distance threshold). Fig. 21a shows the case where number of neighbors is set to 2 and the distance threshold is set to 0.1. This means that only the top 10% minority data points that are closest to their 2nd nearest neighbors are used in the process of generation of artificial points. As a result, the points are extremely concentrated. Fig. 21c shows the opposite extreme where number of neighbors is 7 and the distance threshold is 0.9. This implies that the top 90% minority points closest to their 7th neighbors get to participate in creating the artificial point. The difference between this case and Fig. 21b is in the value of (number of neighbors)k. In Fig. 21b, the minority points used are based on their distances to their 2nd nearest neighbor. Also 2 neighbors are associated in creation of an artificial point whereas in Fig. 21c, 7 minority class neighbors are associated in creating a new point. This is why, although distance threshold

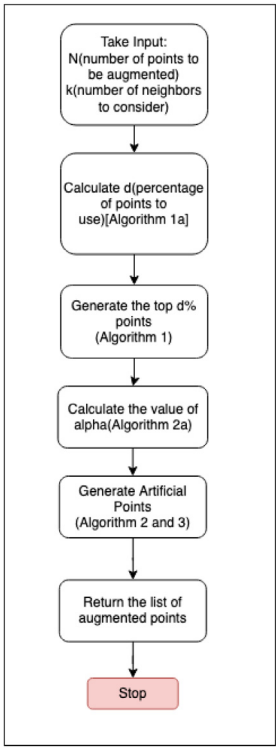

Fig. 18. Flowchart for augmentation of N points considering k neighbors each.

is 0.9 in both Fig. 21 b and c, the spread of augmented data is different. Irrespective of the values of the parameter, the KNNOR algorithm ensures that the selection of original minority points and consequent creation of artificial points is done with as little damage to the original contour of the data as possible.

4. Experiment design

The performance of an oversampler is evaluated in the subsequent classification step. Different classifiers tend to work well with different type of oversamplers [24]. In order to enable fair comparison, the classifiers chosen are k Nearest Neighbors (kNN) [38], Support Vector Machines (SVM) [38], decision trees (DT) [38] and multilayer perceptrons (MLP) [38]. The top ten oversamplers [24] mentioned in Section 2 and KNNOR are coupled with these classifiers for comparison purpose.

4.1. Data

In order to assess the performance of KNNOR the datasets chosen for augmentation and classification are from a commonly accepted set. These are collected from the Knowledge Extraction Based on Evolutionary Learning(KEEL) repository [39] and have been used in recent evaluations [24,40,41]. Many of them are originally from multiclass datasets available in the UCI Machine Learning Repository [19]. The multiclass datasets are rearranged as imbalanced binary classification problem by selecting classes with fewer number of samples to comprise of the minority class and combining the remaining classes to form the majority class. This restructuring is made apparent in the names of the datasets — for example, 'yeast-0-3-5-9_vs_7-8' refers to the UCI database 'yeast' reorganized by using class labels 0, 3, 5, 9 as the (negative) majority class and choosing class labels 7 and 8 to be the (positive) minority class. The datasets have varying Imbalance Ratio(IR) where

$$IR = \frac{N_-}{N_+} \tag{11}$$

N_- is the number of majority (negative) samples and N_+ defines the count of minority (positive) samples. The process of choice of datasets is as follows. At first all the datasets with a population of less than 1000 have been filtered. This gives a collection of 62 datasets. From this set, 19 datasets have been randomly chosen. Table 2 enumerates the name of the datasets, the number of majority–minority samples and the Imbalance Ratio.

Table 2
Different datasets used in comparing KNNOR with other state-of-the-art-algorithms.

Data Set Name	Instances (majority/minority)	Imbalance Ratio
cleveland0vs4 [39]	177(164/13)	12.62
CM1 [42]	498(449/49)	9.16
ecoli0137vs26 [39]	281(274/7)	39.14
ecoli0147vs2356 [39]	336(307/29)	10.59
ecoli034vs5 [39]	200(180/20)	9.0
ecoli0347vs56 [39]	257(232/25)	9.28
ecoli067vs35 [39]	222(200/22)	9.09
ecoli3 [39]	336(301/35)	8.6
glass0123vs456 [39]	214(163/51)	3.2
glass016vs2 [39]	192(175/17)	10.29
glass0 [39]	214(144/70)	2.06
glass1 [39]	214(138/76)	1.82
glass4 [39]	214(201/13)	15.46
habarman [39]	306(225/81)	2.78
newthyroid1 [39]	215(180/35)	5.14
pageblocks13vs4 [39]	472(444/28)	15.86
pima [39]	768(500/268)	1.87
poker9vs7 [39]	244(236/8)	29.5
winequalitywhite3vs7 [39]	900(880/20)	44.0

4.2. Parameter combination for oversamplers and classifiers

The total number of oversamplers were 13. This includes the top 10 oversamplers mentioned in Kovacs [24] work, KNNOR, SMOTE and No Augmentation [24]. SMOTE and No Augmentation have been used as a baseline. Each oversampler has a set of parameter values that can be altered during every experiment. Based on these parameters, 35 combinations have been chosen randomly for model selection. [24]. An upper limit has been put on the number of parameters to maintain a finite runtime. Following the same, [24] 6 different parameterizations have been chosen to evaluate each classifier: **SVM** - C $\in$ {1,10} (performance of SVM improves with high C value in imbalanced data [43]), L1 and L2 penalties with compatible hinge or squared hinge loss; **kNN** - k $\in$ {3, 5, 7}, standard or distance weighted decision functions with L_2 distance; **DT** - Splitting criterion: Gini-impurity/entropy, maximum depth of 3, 5 and unbounded; **MLP** - One hidden layer, activation function: RELU/Logistic, hidden units as 10%, 50% or 100% of training data features

4.3. Cross validation and evaluation

The performance of classifiers is evaluated by repeated stratified k-fold cross-validation with 5 splits and 3 repeats. This is similar to [24]. The evaluation consists of cross-validation involving all the oversamplers with upto 35 random parameter combinations and the 4 classifiers with 6 parameter combinations. Each dataset is folded in the same manner at each test case. The metrics used to compare the results are the G, AUC,

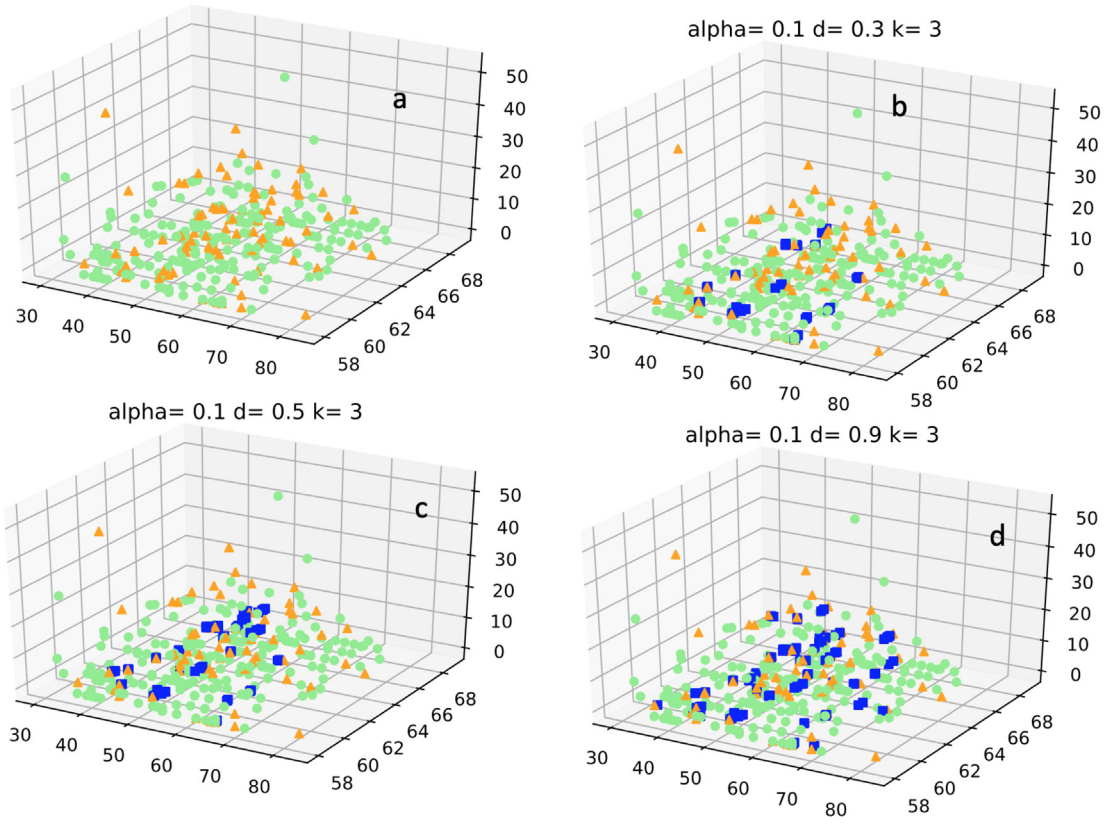

Fig. 19. Creation of 100 artificial points based on different values of the distance threshold for Haberman dataset [19]. Figure (a) shows the original distribution of data. Green squares represent majority, orange triangles represent the minority, and the blue squares depict the augmented data points. $\alpha \epsilon [0, 1]$, d is the percentage of minority points taken in consideration and k is the number of neighbors.

Table 3
Some Notations.

Notation	Meaning	
TP	Number of True positive samples	
FP	Number of False positive samples	
TN	Number of True negative samples	
FN	Number of False negative samples	
Precision	$\frac{TP}{TP+FP}$	[3]
Recall	$\frac{TP}{TP+FN}$	[47]
False Positive Rate	$\frac{FP}{FP+TN}$	[48]

Table 4
Metrics chosen to compare classification performance.

Metric	Definition/Formula	
G-score	$\sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$	[49]
F_1-score	$\frac{2*recall*precision}{(precision)+recall}$	[50]

F1 and P20 scores defined in the next sub section. For each combination of dataset, oversampler and classifier, the top results were generated. This gave a total of 3996 results. These results were then ranked for each set of dataset and classifier and have been discussed in the next section. The volume of the comparative study can be gauged by the fact that the total number of oversampling, training and prediction jobs is more than 2 million.

4.4. Evaluation metrics

Keeping in mind that the data used is imbalanced, the widely accepted and standard measures characterizing binary classification may not be suitable here. In case of imbalanced data, the target is to improve the identification of minority(positive) samples, at the same time conserving the accuracy for the majority class. Following the suggestions of [44–46], 4 performance measures have been chosen. Before defining two of them in Table 4, some notations are enumerated in Table 3.

Other two metrics are, **AUC:** Area under the receiving operating characteristic curve(AUC) is the area under the curve of sensitivities plotted against corresponding false positive rates(FPR) at different threshold values. An intuitive definition is that it is a probabilistic measure of a classifiers tendency to assign a higher positive probability to a random positive sample than assign the same probability to a randomly selected negative sample [24].

Precision at Top 20%(P20): It is the percentage of true positive samples among the top 20 test data points with the maximum probability of belonging to the positive class [44]. In our case, since some of the datasets have less than 20 positive samples, 20% has been used instead of the absolute value of 20.

5. Results

In order to understand how well KNNOR performs in comparison to the other oversamplers, a detailed analysis is presented. The entire result is a part of the additional documents due to page limitations. The experiment is done on 19 different datasets and in order to get a holistic view of the result, the dataset dimension needs to be aggregated from the result. For each performance metric, oversampler type and classifier type, the mean score is computed over all the datasets. The top 9 oversamplers in combination with the various performance metrics and classifiers

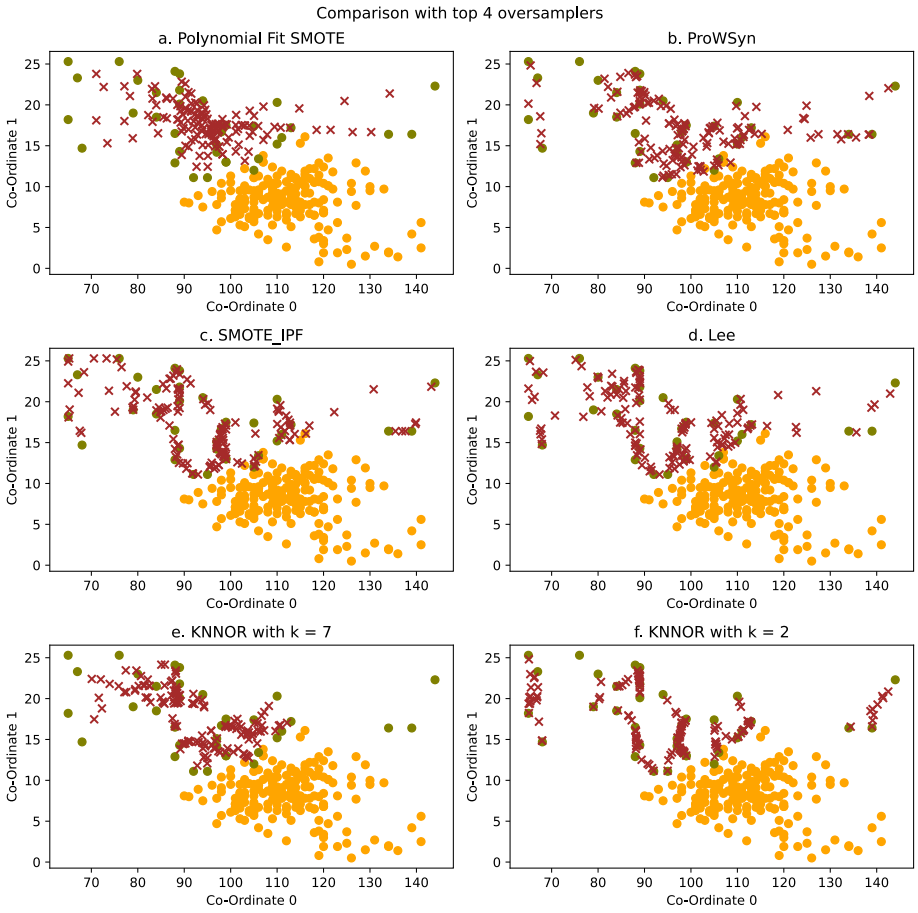

Fig. 20. Creation of artificial points using different oversamplers on the thyroid1 data [19]. Orange and greed dots represent the majority and minority respectively. The red crosses mark the artificial points created.

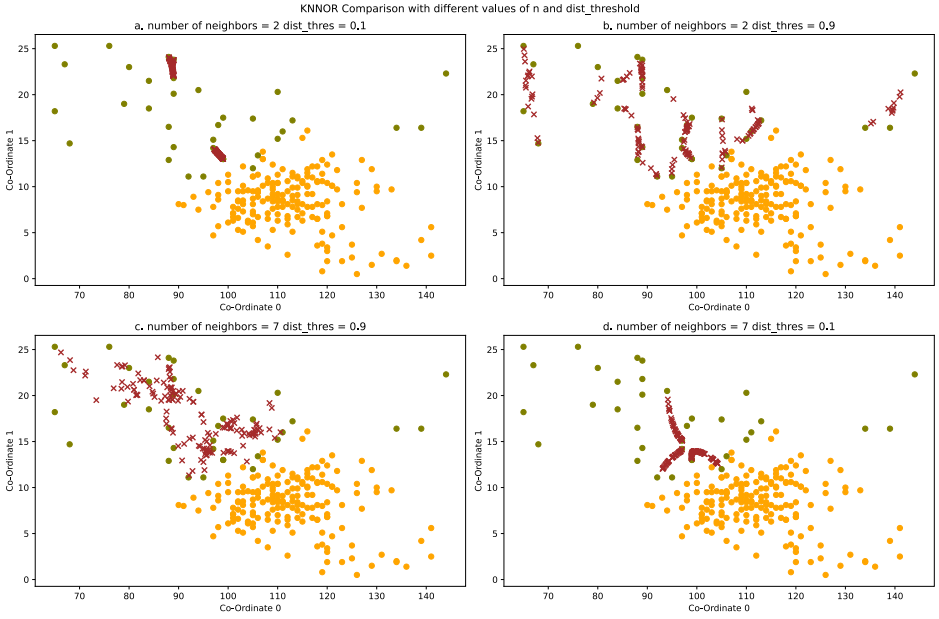

Fig. 21. Creation of artificial points using different parameters of KNNOR on thyroid1 data. Orange and greed dots represent the majority and minority respectively. The red crosses mark the artificial points created.

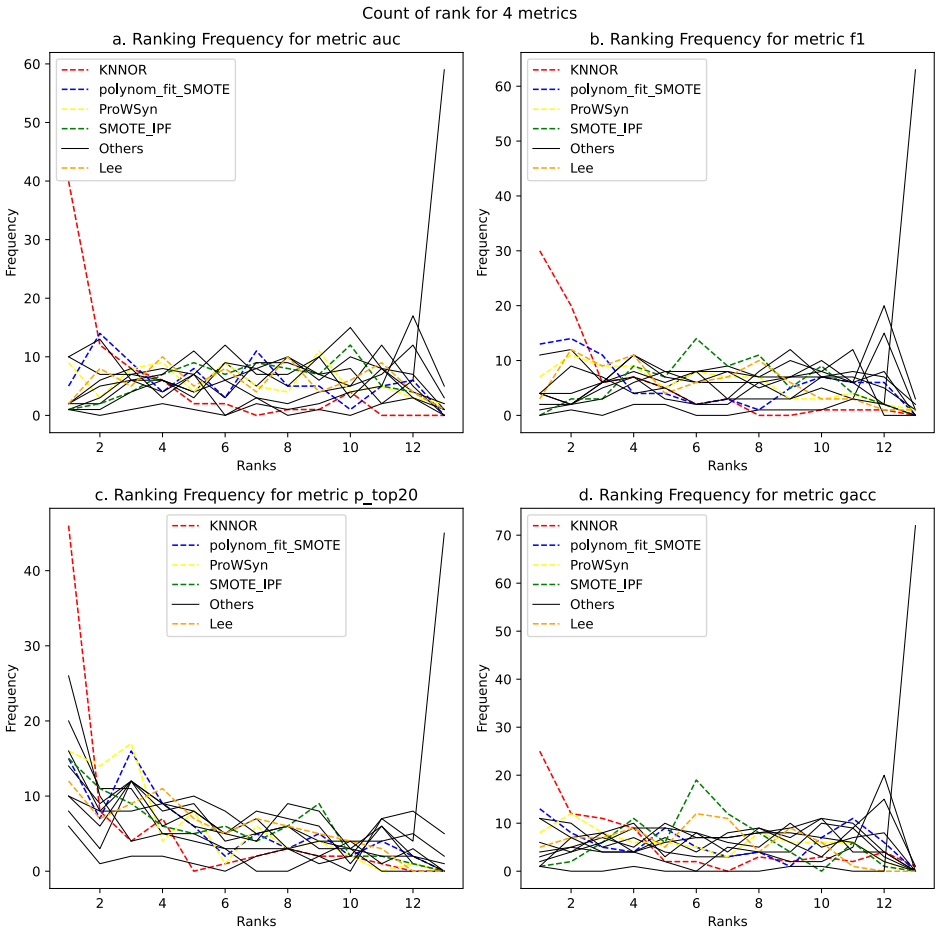

Fig. 22. Frequency of ranks for different oversamplers for each metric. Dotted red line represents KNNOR. The top 4 oversamplers are depicted in yellow(ProWSyn), blue(polynom_fit_SMOTE), green(SMOTE_IPF) and orange(Lee). Other oversamplers are depicted by black lines. The ranks are presented along the X axis starting from Rank1. The vertical axis represents the count of each rank.

have been enlisted in Table 5. BaseLine(BL) scores are kept at the bottom of each list. They comprise of accuracy measures with SMOTE and accuracy without oversampling. KNNOR is found to be standing at the top 3 at all except two tables, implying that it can be used as a reliable oversampling technique for new data. A more comprehensive view of the comparison is given in Table 6 where the results have been aggregated by averaging performance metrics over the classifiers, setting the ranks of the average scores to each oversampler and calculating the mean of ranks. As the meaning of value of metrics varies from measure(F1score) to measure(G score), consideration is given to the average ranks while arranging the top oversamplers. KNNOR is found to top the chart for overall performance in Table 6, proving its effectiveness. Both Tables 5 and 6 are designed following the work of Kovacs [24]. The consistency of KNNOR is also expressed in Fig. 22 where the frequency of ranks for different oversamplers are plotted. It shows that KNNOR has appeared at the 1st rank more than any other oversamplers. The top 4 state-of-art oversamplers, on the other hand have not been able to maintain the consistency as well as KNNOR.

These figures bear testimony that KNNOR can be considered to be a good oversampler candidate for novel imbalanced data. The code to perform the oversampling, classification and comparison has been used from the repository provided in [37]. The authors forked the repository and added the code of KNNOR to keep the results compatible for comparison. The entire code base for comparison can be found at the github repository [51].

6. Image data

KNNOR is also applied on Image data to check its performance on high dimension data. The algorithm was applied on UTK-Face [52] data by keeping a small set of male faces as the minority and a large number of female faces as majority set. The algorithm was also applied to the CIFAR-10 dataset [53], keeping pictures of airplanes in the minority set and the remaining images in the majority set. Fig. 23 shows the artificial images generated, using the pre existing face images already present. As the generated point is derived from a minority point and its neighbors, the images bear similarity to existing images. Fig. 24 shows artificial airplane images generated using KNNOR. Depending on the parameters chosen for KNNOR, the images can be made to be more dissimilar from existing images with the risk of disfiguring the image.

7. Conclusion

In this paper we propose a generalized augmentation method (KNNOR) for imbalanced data classification. Based on the original data distribution, KNNOR can find out the critical areas of augmentation and synthesize datapoints more effectively. This is accomplished by finding clusters of points which are closer to their neighbors and by giving them higher priority. The points which are farther apart are not totally ignored but have a lower importance in generation of points. This enables the algorithm to perform better than the existing approaches. The method has been implemented as a python library and is available at [54].

Table 5
The top performing oversamplers based on the average scores for all datasets: the four sets of rows correspond to the four metrics chosen(AUC, F1-score, G-score and P20) and the columns denote classification techniques. Each cell contains the top oversampling techniques ranked in descending order. BL implies BaseLine.

Classifier	SVM		DT		KNN		MLP	
Rank	Sampler	AUC	Sampler	AUC	Sampler	AUC	Sampler	AUC
1	*KNNOR*	.8892	LVQ-SMOTE [31]	.8686	polynom-fit-SMOTE [25]	.905	*KNNOR*	.9055
2	G-SMOTE [29]	.8815	*KNNOR*	.8594	ProWSyn [26]	.9043	polynom-fit-SMOTE [25]	.8899
3	polynom-fit-SMOTE [25]	.8729	CCR [30]	.8523	Assembled SMOTE [32]	.903	ProWSyn [26]	.8884
4	CCR [30]	.8699	polynom-fit-SMOTE [25]	.8483	Lee [27]	.9013	Lee [27]	.8884
5	ProWSyn [26]	.8699	SMOBD [28]	.8453	*KNNOR*	.9011	Assembled SMOTE [32]	.8879
6	SMOTE - Tomeklinks [33]	.8684	ProWSyn [26]	.842	SMOTE-IPF [22]	.9004	CCR [30]	.8878
7	SMOTE-IPF [22]	.8676	SMOTE [18]	.8416	SMOBD [28]	.9001	SMOBD [28]	.8875
8	SMOBD [28]	.8675	G-SMOTE [29]	.8408	SMOTE - Tomeklinks [33]	.8993	G-SMOTE [29]	.8869
9	Lee [27]	.8671	Lee [27]	.8408	SMOTE [18]	.8989	SMOTE-TomekLinks [33]	.885
BL	SMOTE [18]	.8666	SMOTE [18]	.8416	SMOTE [18]	.8989	SMOTE [18]	.8842
BL	No Oversampling	.8627	No Oversampling	.8075	No Oversampling	.8773	No Oversampling	.8097
Rank	Sampler	F1	Sampler	F1	Sampler	F1	Sampler	F1
1	polynom-fit-SMOTE [25]	.6845	*KNNOR*	.7051	*KNNOR*	.747	*KNNOR*	.6869
2	*KNNOR*	.6775	CCR [30]	.6865	polynom-fit-SMOTE [25]	.7468	CCR [30]	.6669
3	Assembled SMOTE [32]	.6763	LVQ-SMOTE [31]	.6812	CCR [30]	.7391	polynom-fit-SMOTE [25]	.6657
4	ProWSyn [26]	.675	G-SMOTE [29]	.6756	ProWSyn [26]	.7239	Lee [27]	.6583
5	G-SMOTE [29]	.6749	ProWSyn [26]	.672	LVQ-SMOTE [31]	.7227	ProWSyn [26]	.656
6	Lee [27]	.6739	Assembled SMOTE [32]	.6717	G-SMOTE [29]	.7219	SMOTE - Tomeklinks [33]	.6524
7	SMOBD [28]	.6707	Lee [27]	.6714	Lee [27]	.7219	G-SMOTE [29]	.6506
8	SMOTE - Tomeklinks [33]	.6699	SMOTE-IPF [22]	.6712	SMOBD [28]	.7202	SMOTE-IPF [22]	.6502
9	SMOTE-IPF [22]	.6693	polynom-fit-SMOTE [25]	.6694	SMOTE-IPF [22]	.7195	SMOBD [28]	.6497
BL	SMOTE [18]	.6643	SMOTE [18]	.665	SMOTE [18]	.7163	SMOTE [18]	.6486
BL	No Oversampling	.4421	No Oversampling	.6172	No Oversampling	.6787	No Oversampling	.4273
Rank	Sampler	P20	Sampler	P20	Sampler	P20	Sampler	P20
1	*KNNOR*	.4481	LVQ-SMOTE [31]	.4404	*KNNOR*	.4705	*KNNOR*	.4605
2	G-SMOTE [29]	.4377	*KNNOR*	.4384	ProWSyn [26]	.4678	ProWSyn [26]	.4468
3	SMOBD [28]	.4373	ProWSyn [26]	.4283	polynom-fit-SMOTE [25]	.4659	SMOBD [28]	.4442
4	SMOTE - Tomeklinks [33]	.437	SMOBD [28]	.4277	LVQ-SMOTE [31]	.4648	polynom-fit-SMOTE [25]	.4439
5	polynom-fit-SMOTE [25]	.4363	polynom-fit-SMOTE [25]	.4268	CCR [30]	.4645	Assembled SMOTE [32]	.4434
6	ProWSyn [26]	.436	CCR [30]	.4266	SMOTE-IPF [22]	.4645	Lee [27]	.4434
7	Assembled SMOTE [32]	.4358	SMOTE-IPF [22]	.4252	Assembled SMOTE [32]	.4634	SMOTE [18]	.4426
8	LVQ-SMOTE [31]	.4351	Lee [27]	.4235	SMOBD [28]	.4626	SMOTE-IPF [22]	.4425
9	SMOTE-IPF [22]	.4348	Assembled SMOTE [32]	.423	Lee [27]	.4624	G-SMOTE [29]	.4405
BL	SMOTE [18]	.4336	SMOTE [18]	.4225	SMOTE [18]	.4602	SMOTE [18]	.4426
BL	No Oversampling	.4185	No Oversampling	.3974	No Oversampling	.4516	No Oversampling	.3512
Rank	Sampler	G	Sampler	G	Sampler	G	Sampler	G
1	*KNNOR*	.8363	LVQ-SMOTE [31]	.8356	polynom-fit-SMOTE [25]	.8611	*KNNOR*	.8508
2	polynom-fit-SMOTE [25]	.834	*KNNOR*	.823	Assembled SMOTE [32]	.8592	ProWSyn [26]	.8428
3	CCR [30]	.8324	SMOTE [18]	.8198	SMOTE [18]	.858	Lee [27]	.8399
4	Assembled SMOTE [32]	.8282	G-SMOTE [29]	.8179	SMOTE - Tomeklinks [33]	.8572	SMOBD [28]	.8373
5	ProWSyn [26]	.8273	ProWSyn [26]	.8175	Lee [27]	.8572	polynom-fit-SMOTE [25]	.8371
6	Lee [27]	.8271	SMOBD [28]	.8162	SMOTE-IPF [22]	.8568	SMOTE-IPF [22]	.8367
7	SMOTE - Tomeklinks [33]	.8269	SMOTE-IPF [22]	.8162	ProWSyn [26]	.8556	G-SMOTE [29]	.8364
8	SMOBD [28]	.8263	Lee [27]	.8147	SMOBD [28]	.8544	Assembled SMOTE [32]	.8364
9	G-SMOTE [29]	.8252	Assembled SMOTE [32]	.8146	*KNNOR*	.8512	SMOTE [18]	.8342
BL	SMOTE [18]	.8238	SMOTE [18]	.8198	SMOTE [18]	.858	SMOTE [18]	.8342
BL	No Oversampling	.529	No Oversampling	.7432	No Oversampling	.7525	No Oversampling	.6114

Table 6
Ranking oversamplers based on a combination of all scores. Apart from the combined rank, the aggregated values of the metrics and the subsequent ranks are also documented.

Rank	Sampler	Average Rank	AUC	AUC Rank	G	G Rank	F1	F1 Rank	P20	P20 Rank
1	KNNOR	1	0.8888	1	0.7041	1	0.4544	1	0.8403	1
2	CCR [30]	7.5	0.8765	4	0.6903	3	0.4402	11	0.8295	12
3	polynom-fit-SMOTE [25]	2.5	0.879	2	0.6916	2	0.4432	4	0.836	2
4	SMOTE-TomekLinks [33]	10.25	0.8721	12	0.6758	10	0.4403	10	0.8317	9
5	ProWSyn [26]	3.5	0.8761	5	0.6817	4	0.4447	2	0.8358	3
6	SMOBD [28]	7.25	0.8751	7	0.6761	9	0.443	5	0.8336	8
7	SMOTE-IPF [22]	7.75	0.8733	10	0.6776	8	0.4417	6	0.8337	7
8	G-SMOTE [29]	7.75	0.8757	6	0.6807	6	0.4406	9	0.8313	10
9	SMOTE [18]	10.25	0.8728	11	0.6736	12	0.4397	12	0.834	6
10	Assembled-SMOTE [32]	6.75	0.8744	8	0.6789	7	0.4414	7	0.8346	5

Fig. 23. Artificial faces generated by KNNOR.

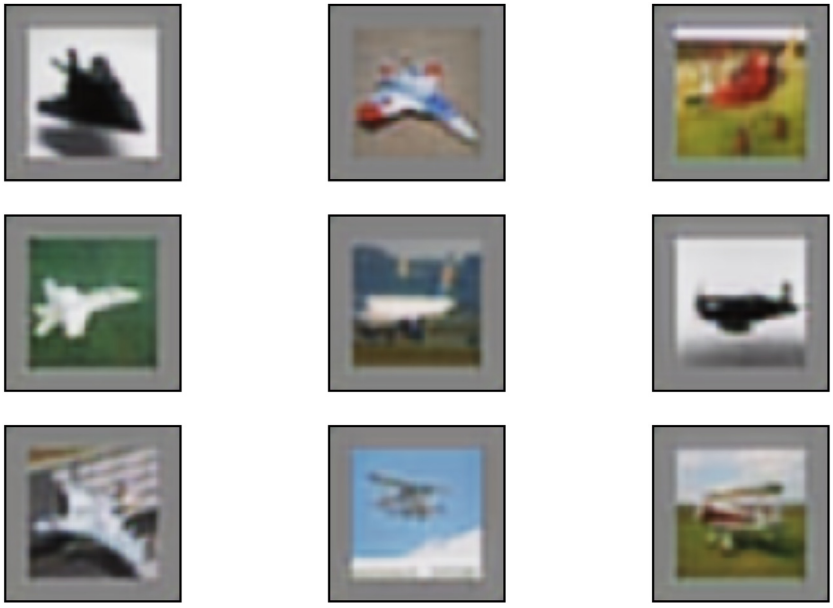

Fig. 24. Artificial CIFAR10 airplane images generated by KNNOR.

The documentation for the library is available at [55]. In future the same method of augmentation can be applied to multi-class classification by applying the algorithm on each of the minority datasets. The procedure can also be applied to streaming data, an example of which can be found in [56]. This data driven method can also be coupled with other algorithm driven approaches like ensemble support vector machines or extreme learning to attain even better results.

CRediT authorship contribution statement

Ashhadul Islam: Methodology, Software, Writing – original draft preparation & editing. **Samir Brahim Belhaouari:** Conceptualization, Supervision, Review. **Atiq Ur Rehman:** Writing – review & editing, Validation. **Halima Bensmail:** Methodology, Resources, Review.

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.

Acknowledgements

Open Access funding provided by the Qatar National Library.

References

[1] M. Eshtay, H. Faris, N. Obeid, Improving extreme learning machine by competitive swarm optimization and its application for medical diagnosis problems, Expert Syst. Appl. 104 (2018) 134–152, http://dx.doi.org/10.1016/j.eswa.2018.03.024.

[2] S.V. Kovalchuk, E. Krotov, P.A. Smirnov, D.A. Nasonov, A.N. Yakovlev, Distributed data-driven platform for urgent decision making in cardiological ambulance control, Future Gener. Comput. Syst. 79 (2018) 144–154, http://dx.doi.org/10.1016/j.future.2016.09.017.

[3] R. Nagarajan, M. Upreti, An ensemble predictive modeling framework for breast cancer classification, Methods 131 (July) (2017) 128–134, http://dx.doi.org/10.1016/j.ymeth.2017.07.011.

[4] D. Gan, J. Shen, B. An, M. Xu, N. Liu, Integrating TANBN with cost sensitive classification algorithm for imbalanced data in medical diagnosis, Comput. Ind. Eng. 140 (June 2019) (2020) 106266, http://dx.doi.org/10.1016/j.cie.2019.106266.

[5] B. Krawczyk, M. Woźniak, G. Schaefer, Cost-sensitive decision tree ensembles for effective imbalanced classification, Appl. Soft Comput. 14 (PART C) (2014) 554–562, http://dx.doi.org/10.1016/j.asoc.2013.08.014.

[6] N. Liu, J. Shen, M. Xu, D. Gan, E.S. Qi, B. Gao, Improved cost-sensitive support vector machine classifier for breast cancer diagnosis, Math. Probl. Eng. 2018 (2018) http://dx.doi.org/10.1155/2018/3875082.

[7] Y. Liu, X. Yu, J.X. Huang, A. An, Combining integrated sampling with SVM ensembles for learning from imbalanced datasets, Inf. Process. Manage. 47 (4) (2011) 617–631, http://dx.doi.org/10.1016/j.ipm.2010.11.007.

[8] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybernet. C Appl. Rev. 42 (4) (2012) 463–484, http://dx.doi.org/10.1109/TSMCC.2011.2161285.

[9] R.C. Holte, L. Acker, B. Porter, Concept learning and the problem of small disjuncts, in: Ijcai, 1989, pp. 813–818.

[10] R. Laza, R. Pavón, M. Reboiro-Jato, F. Fdez-Riverola, Evaluating the effect of unbalanced data in biomedical document classification, J. Integr. Bioinform. 8 (3) (2011) 177, http://dx.doi.org/10.1515/jib-2011-177.

[11] E. AT, A. M, A.M. F, S. M, Classification of imbalance data using tomek link (T-Link) Combined with random under-sampling (RUS) as a data reduction method, Glob. J. Technol. Optim. 01 (S1) (2016) 1–11, http://dx.doi.org/10.4172/2229-8711.s1111.

[12] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selectio, in: International Conference on Machine Learning, Vol. 4, 1997, pp. 186–197.

[13] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2101, 2001, pp. 63–66, http://dx.doi.org/10.1007/3-540-48229-6_9.

[14] P. Thanathamathee, C. Lursinsap, Handling imbalanced data sets with synthetic boundary data generation using bootstrap re-sampling and adaboost techniques, in: Pattern Recognition Letters, Vol. 34, Elsevier B.V., 2013, pp. 1339–1347, http://dx.doi.org/10.1016/j.patrec.2013.04.019.

[15] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (1) (2019) http://dx.doi.org/10.1186/s40537-019-0192-5, URL: https://doi.org/10.1186/s40537-019-0192-5.

[16] H. He, E. Garcia, Learning from imbalanced data, in: Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, Vol. 21, No. 9, 2009, pp. 923–930, http://dx.doi.org/10.1109/ICTAI.2019.00131, arXiv:1909.01651.

[17] X.Y. Liu, Z.Z. J. Wu, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. 39 (2) (2009) 539–550, http://dx.doi.org/10.1007/978-3-642-34478-7_71.

[18] W.P.K. Nitesh V. Chawla Kevin W. Bowyer, Lawrence O. Hall, SMOTE: SYnthetic minority over-sampling technique, J. Artificial Intelligence Res. (2002) 321–357, arXiv:1106.1813.

[19] UCI, UCI Machine learning laboratory datasets, URL: https://archive.ics.uci.edu/ml/datasets.php.

[20] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5476 LNAI, 2009, pp. 475–482, http://dx.doi.org/10.1007/978-3-642-01307-2_43.

[21] W.A. Rivera, Noise reduction a priori synthetic over-sampling for class imbalanced data sets, Inform. Sci. 408 (2017) 146–161, http://dx.doi.org/10.1016/j.ins.2017.04.046.

[22] J.A. Sáez, J. Luengo, J. Stefanowski, F. Herrera, SMOTE-IPF: ADdressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, Inform. Sci. 291 (C) (2015) 184–203, http://dx.doi.org/10.1016/j.ins.2014.08.051.

[23] B. Zhu, B. Baesens, S.K. vanden Broucke, An empirical comparison of techniques for the class imbalance problem in churn prediction, Inform. Sci. 408 (2017) 84–99, http://dx.doi.org/10.1016/j.ins.2017.04.015.

[24] G. Kovács, An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets, Appl. Soft Comput. 83 (July) (2019) http://dx.doi.org/10.1016/j.asoc.2019.105662.

[25] S. Gazzah, N.E.B. Amara, New oversampling approaches based on polynomial fitting for imbalanced data sets, in: DAS 2008 - Proceedings of the 8th IAPR International Workshop on Document Analysis Systems, 2008, pp. 677–684, http://dx.doi.org/10.1109/DAS.2008.74.

[26] S. Barua, M.M. Islam, K. Murase, ProWSyn: PRoximity weighted synthetic oversampling technique for imbalanced data set learning, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7819 LNAI, (PART 2) 2013, pp. 317–328, http://dx.doi.org/10.1007/978-3-642-37456-2_27.

[27] J. Lee, N.R. Kim, J.H. Lee, An over-sampling technique with rejection for imbalanced class learning, in: ACM IMCOM 2015 - Proceedings, 2015, http://dx.doi.org/10.1145/2701126.2701181.

[28] Q. Cao, S. Wang, Applying over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning, in: Proceedings - 2011 4th International Conference on Information Management, Innovation Management and Industrial Engineering, ICIII 2011, Vol. 2, IEEE, 2011, pp. 543–548, http://dx.doi.org/10.1109/ICIII.2011.276.

[29] T. Sandhan, J.Y. Choi, Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition, in: Proceedings - International Conference on Pattern Recognition, (August) 2014, pp. 1449–1453, http://dx.doi.org/10.1109/ICPR.2014.258.

[30] M. Koziarski, M. Wozniak, CCR: A Combined cleaning and resampling algorithm for imbalanced data classification, Int. J. Appl. Math. Comput. Sci. 27 (4) (2017) 727–736, http://dx.doi.org/10.1515/amcs-2017-0050.

[31] M. Nakamura, Y. Kajiwara, A. Otsuka, H. Kimura, LVQ-SMOTE - Learning vector quantization based synthetic minority over-sampling technique for biomedical data, BioData Min. 6 (1) (2013) 1–10, http://dx.doi.org/10.1186/1756-0381-6-16.

[32] B. Zhou, C. Yang, H. Guo, J. Hu, A quasi-linear SVM combined with assembled SMOTE for imbalanced data classification, in: Proceedings of the International Joint Conference on Neural Networks, IEEE, 2013, http://dx.doi.org/10.1109/IJCNN.2013.6707035.

[33] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 20–29, http://dx.doi.org/10.1145/1007730.1007735.

[34] A. Savitzky, G. Marcel, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (8) (1964) 1627–1639, http://dx.doi.org/10.1021/ac60214a048.

[35] G.L. Libralon, A.C.L.F. Carvalho, A.C. Lorena, Ensembles of pre-processing techniques for noise detection in gene expression data, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5506 LNCS, (PART 1) 2009, pp. 486–493, http://dx.doi.org/10.1007/978-3-642-02490-0_60.

[36] A. Böttcher, D. Wenzel, The frobenius norm and the commutator, Linear Algebra Appl. 429 (8–9) (2008) 1864–1885, http://dx.doi.org/10.1016/j.laa.2008.05.020.

[37] G. Kovács, Smote-variants: A python implementation of 85 minority oversampling techniques, Neurocomputing 366 (2019) 352–354, http://dx.doi.org/10.1016/j.neucom.2019.06.100.

[38] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, in: Understanding Machine Learning: From Theory to Algorithms, 9781107057, 2013, pp. 1–397, http://dx.doi.org/10.1017/CBO9781107298019.

[39] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL Data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.

[40] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Inform. Sci. 250 (2013) 113–141, http://dx.doi.org/10.1016/j.ins.2013.07.007.

[41] V. López, A. Fernández, F. Herrera, On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed, Inform. Sci. 257 (2014) 1–13, http://dx.doi.org/10.1016/j.ins.2013.09.038.

[42] X. Zhang, Y. Li, R. Kotagiri, L. Wu, Z. Tari, M. Cheriet, KRNN: K rare-class nearest neighbour classification, Pattern Recognit. 62 (2017) 33–44, http://dx.doi.org/10.1016/j.patcog.2016.08.023.

[43] M.A. Farquad, I. Bose, Preprocessing unbalanced data using support vector machine, Decis. Support Syst. 53 (1) (2012) 226–233, http://dx.doi.org/10.1016/j.dss.2012.01.016.

[44] J. De La Calleja, O. Fuentes, A distance-based over-sampling method for learning from imbalanced data sets, in: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2007, January, 2007, pp. 634–635.

[45] J. De La Calleja, O. Fuentes, J. González, Selecting minority examples from misclassified data for over-sampling, in: Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference, FLAIRS-21, January, 2008, pp. 276–281.

[46] P. Cao, X. Liu, J. Zhang, D. Zhao, M. Huang, O. Zaiane, 2,1 Norm regularized multi-kernel based joint nonlinear feature selection and over-sampling for imbalanced data classification, Neurocomputing 234 (September) (2017) 38–57, http://dx.doi.org/10.1016/j.neucom.2016.12.036.
[47] J.P. Li, A.U. Haq, S.U. Din, J. Khan, A. Khan, A. Saboor, Heart disease identification method using machine learning classification in E-healthcare, IEEE Access 8 (2020) 107562–107582, http://dx.doi.org/10.1109/ACCESS.2020.3001149.
[48] S. Barua, M.M. Islam, K. Murase, A novel synthetic minority oversampling technique for imbalanced data set learning, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7063 LNCS, (PART 2) 2011, pp. 735–744, http://dx.doi.org/10.1007/978-3-642-24958-7_85.
[49] Y. Kazemi, S.A. Mirroshandel, A novel method for predicting kidney stone type using ensemble learning, Artif. Intell. Med. 84 (2018) 117–126, http://dx.doi.org/10.1016/j.artmed.2017.12.001.
[50] H. Wang, B. Zheng, S.W. Yoon, H.S. Ko, A support vector machine-based ensemble algorithm for breast cancer diagnosis, European J. Oper. Res. 267 (2) (2018) 687–699, http://dx.doi.org/10.1016/j.ejor.2017.12.001.
[51] A. Islam, S. Belhaouari, Github repository for comparison code, 2021, URL: https://github.com/ashhadulislam/smote{_}variants.
[52] Z. Zhang, Y. Song, H. Qi, Age progression/regression by conditional adversarial autoencoder, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua, 2017, pp. 4352–4360, http://dx.doi.org/10.1109/CVPR.2017.463, arXiv:1702.08423.
[53] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009.
[54] A. Islam, S. Belhaouari, AugmentData KNNOR, URL: https://pypi.org/project/augmentdata/.
[55] A. Islam, AugmentData KNNOR manual docs, URL: https://augmentdatalib-docs.readthedocs.io/en/latest/.
[56] A. Islam, Streaming credit card data github, 2020.

Ashhadul Islam obtained the master's degree in Information Technology from the International Institute of Information Technology, Bengaluru, India, in 2017. He is currently pursuing the Ph.D. degree in computer science and engineering with Hamad Bin Khalifa University, Qatar. His research interests include data mining and machine learning algorithms. He is proficient in building web based deep learning applications and has more than 3 years of working experience in Development Bank Of Singapore as a Data Scientist.

Samir Brahim Belhaouari (Senior Member, IEEE) received the master's degree in Telecommunications from the National Polytechnic Institute (ENSEEIHT) of Toulouse, France, in 2000, and the Ph.D. degree in applied mathematics from the Federal Polytechnic School of Lausanne (EPFL), in 2006. He is presently an Associate Professor at the Division of Information and Communication Technologies, College of Science and Engineering, HBKU. Several academic and administrator positions are also held by him. These include the vice dean for academic and student affairs with the College of Science and General Studies and University Preparatory Program at ALFAISAL University (KSA), University of Sharjah (UAE), Innopolis University, Russia, Petronas University(Malaysia), and EPFL Federal Swiss School, Switzerland. Machine learning and number theory are his main research interests, with a proclivity to stochastic processes. He is currently involved in the development of algorithms in machine learning, applied to visual surveillance and biomedical data. He has also acquired the support of several international research funds in Russia, Malaysia, and GCC.

Atiq Ur Rehman (Member, IEEE) received the master's degree in computer engineering from the National University of Sciences and Technology(NUST), Pakistan, in 2013, and the Ph.D. degree in computer science and engineering from Hamad Bin Khalifa University, Qatar, in 2019. He is currently working as a Postdoctoral Researcher with the College of Science and Engineering, Hamad Bin Khalifa University. His research interests include the development of evolutionary computation, pattern recognition, and machine learning algorithms.

Halima Bensmail is an Associate Professor at HBKU's College of Public Health, a Principal Scientist of Biostatistics/Statistics and Computational Biology at the Qatar Computing Research Institute (QCRI), and founder of the Computational Biology group (CS&E) at QCRI. Dr. Bensmail served as an Associate Professor at the University of Virginia (EVMS) and an Adjunct Faculty member at Oak Ridge National Laboratory, USA. Her work focuses on developing novel machine learning algorithms to solve large-scale and high-dimensional problems in order to achieve significant biomedical discoveries.