A Resampling Approach for Binary Imbalanced Classification

Dongeun Min[1], Hae Hwan Lee[1], Jongho Im[1,2]
1 Department of Statistics and Data Science, Yonsei University
2 Department of Applied Statistics, Yonsei University

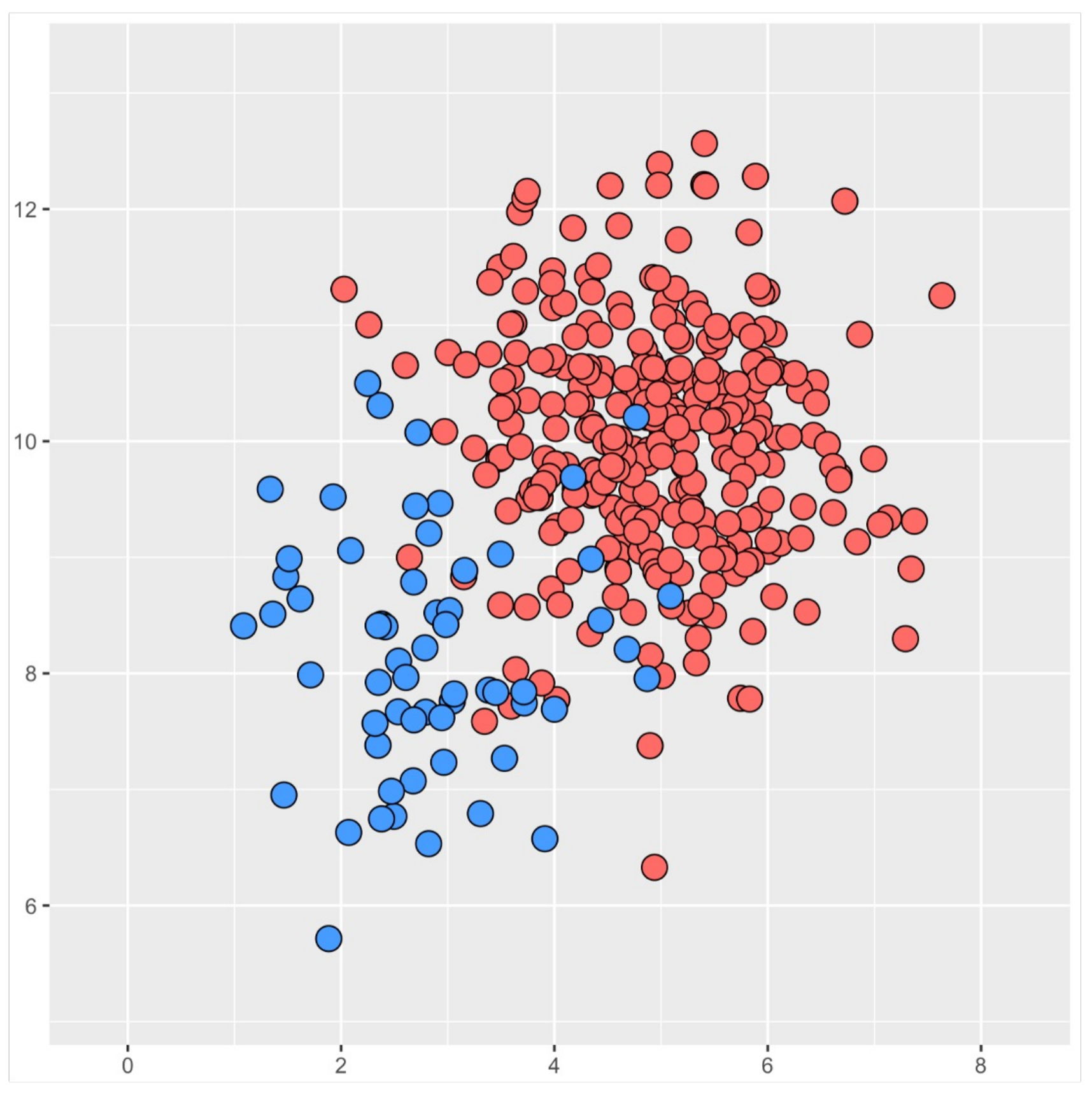
Outline

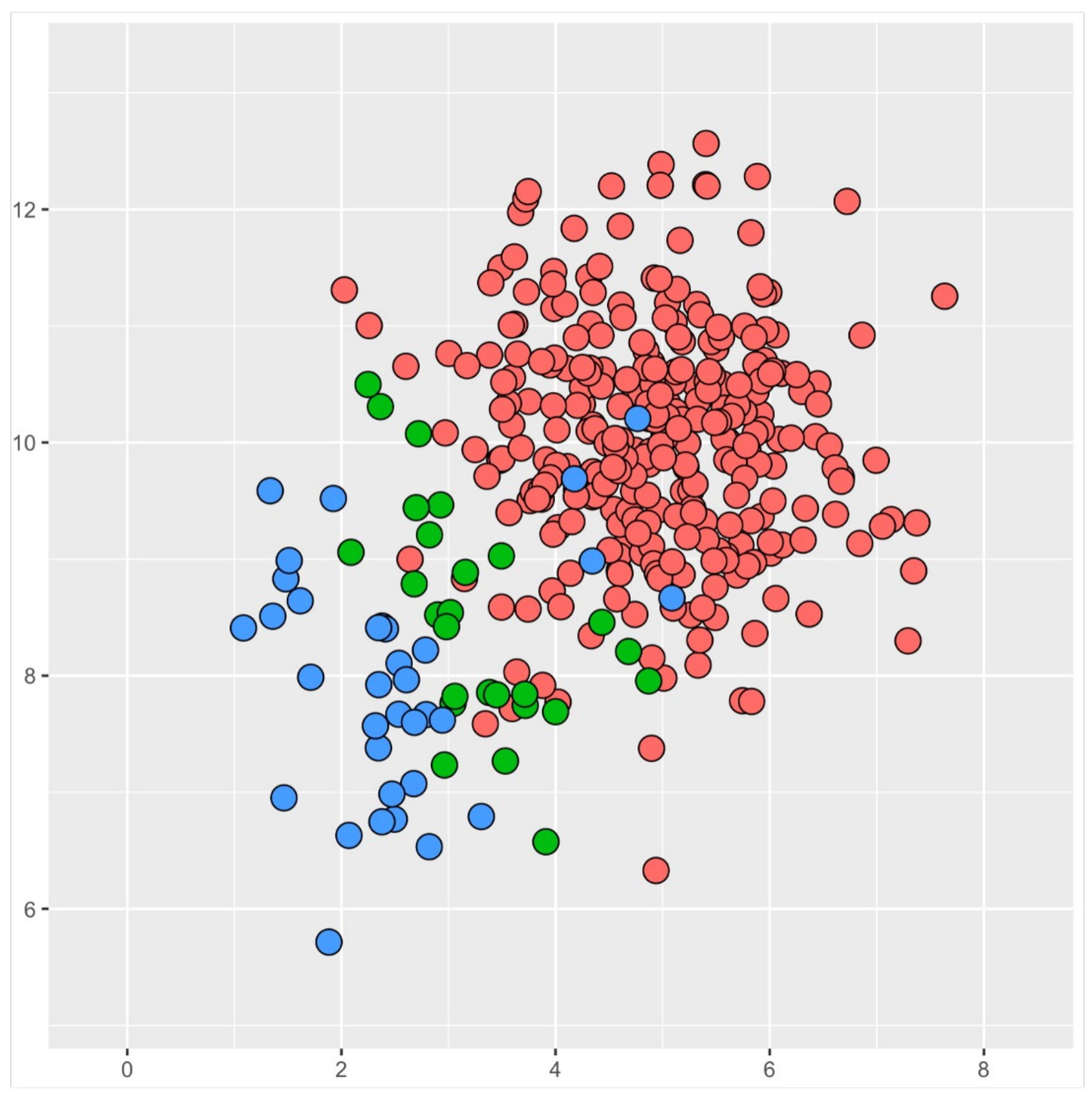
Imbalanced Classification

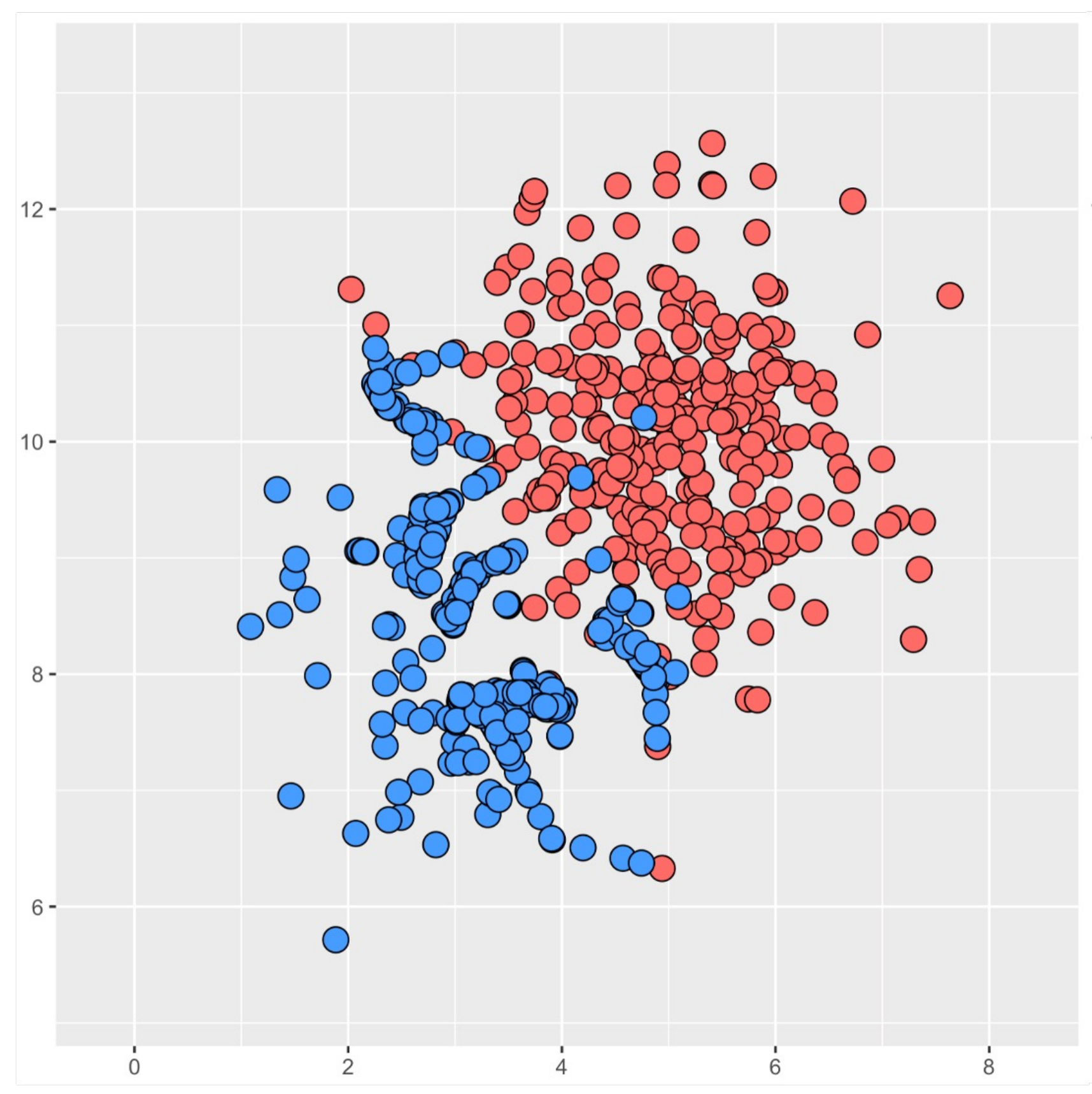
- Imbalanced classification: Classification of data where one class consists of a smaller number of samples than the other class (or classes)
- We aim to propose a sampling-based approach for binary imbalanced classification
- Inspired by Borderline-SMOTE(Han et al., 2005) and RnR(Park et al., 2022), our proposed method generates synthetic samples within the overlapping area where minority samples have majority neighbors and filter out noisy synthetic samples using the original minority sample mean.

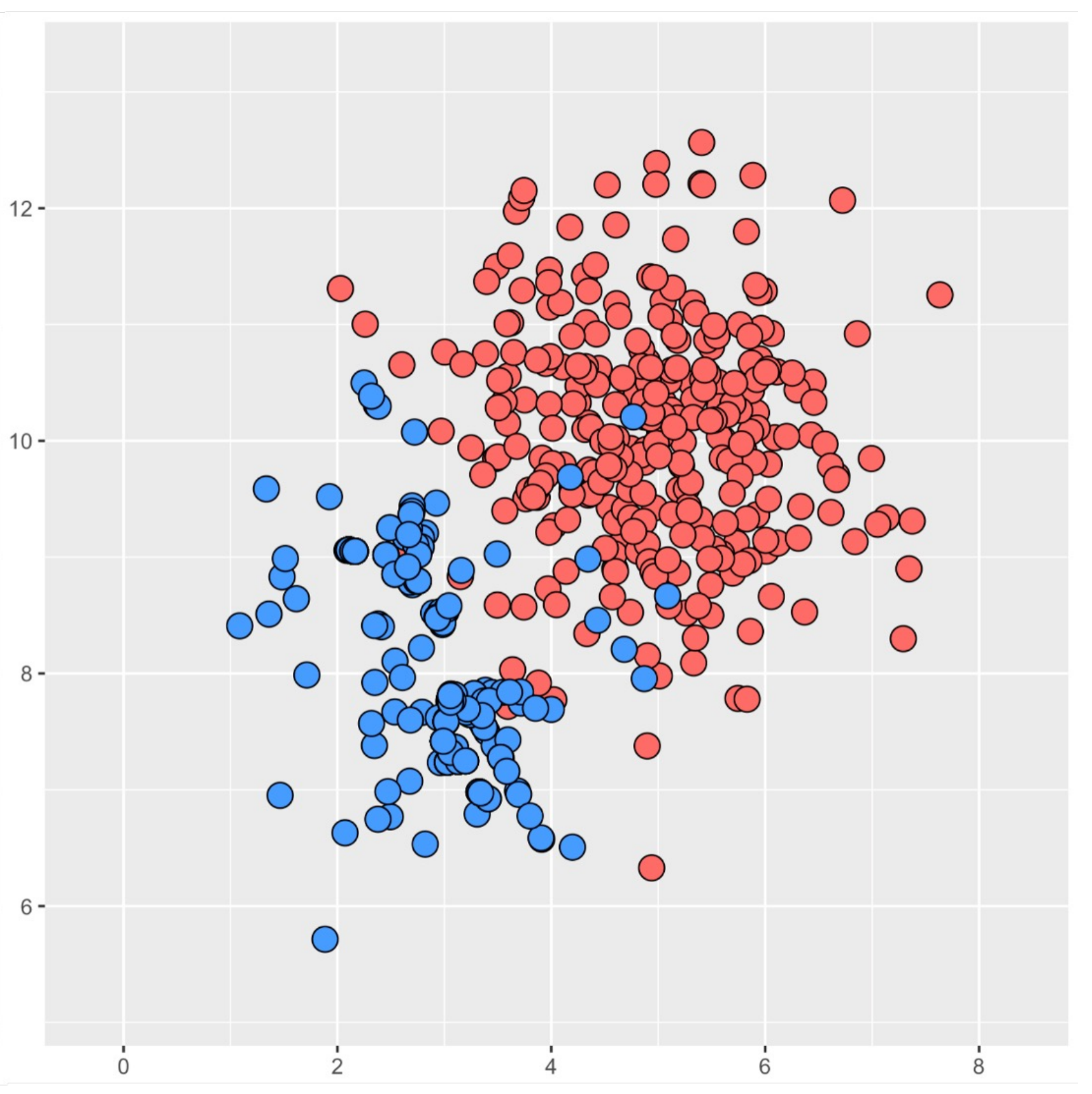
Proposed Method

The proposed method consists of **three steps: 1) Overlap Detection, 2) Synthetic Sample Generation, and 3) Filtering Synthetic Samples**


1. Original Sample


2. Overlap Detection


3. Synthetic Sample Generation

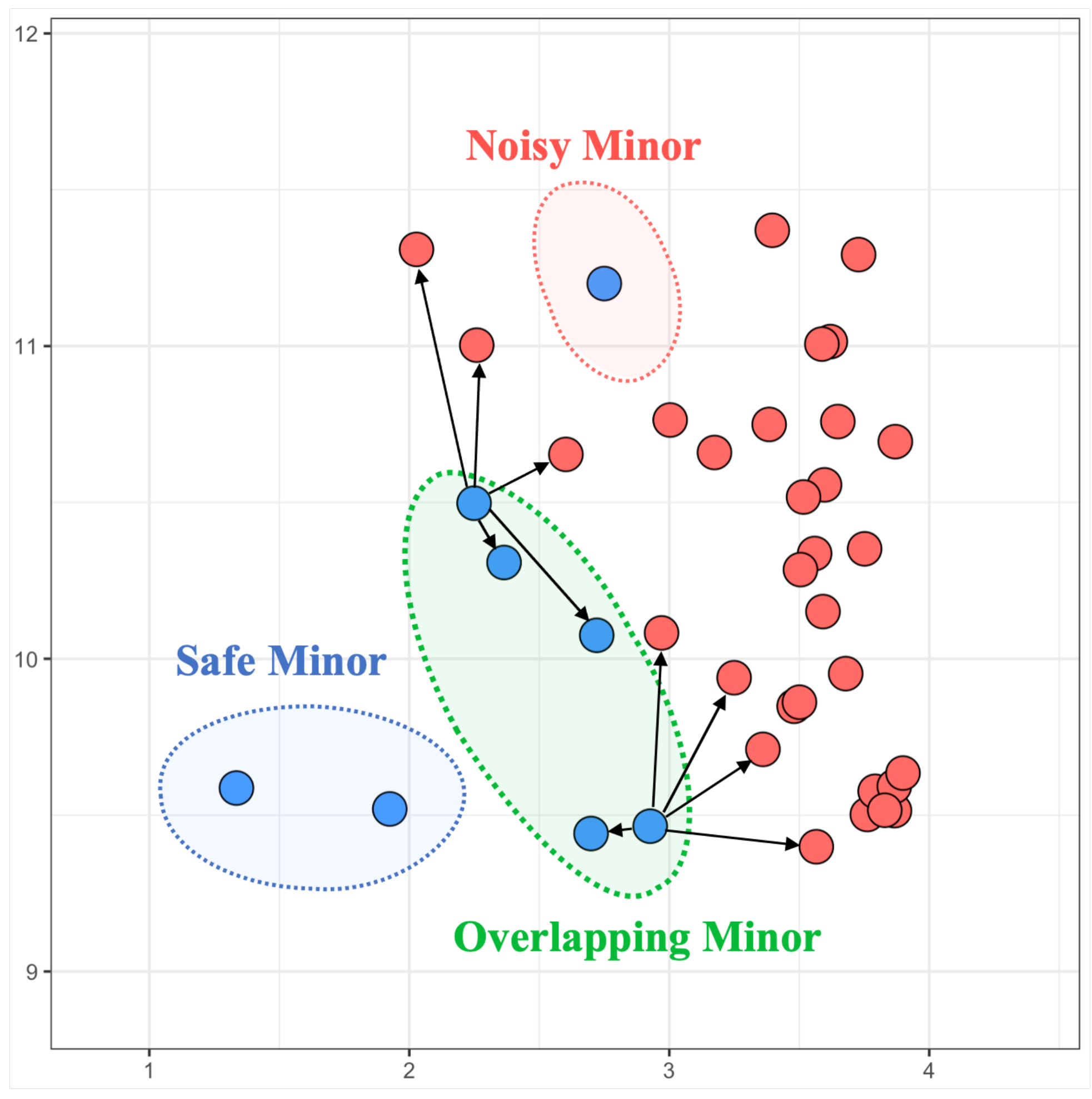

4. Filtering Synthetic Samples

n_1 majority samples, n_2 minority samples ($n_1 > n_2$)

1. Overlap Detection

○ Detect minority samples whose k-nearest neighbors are from both the majority class and minority class as overlapping minority samples

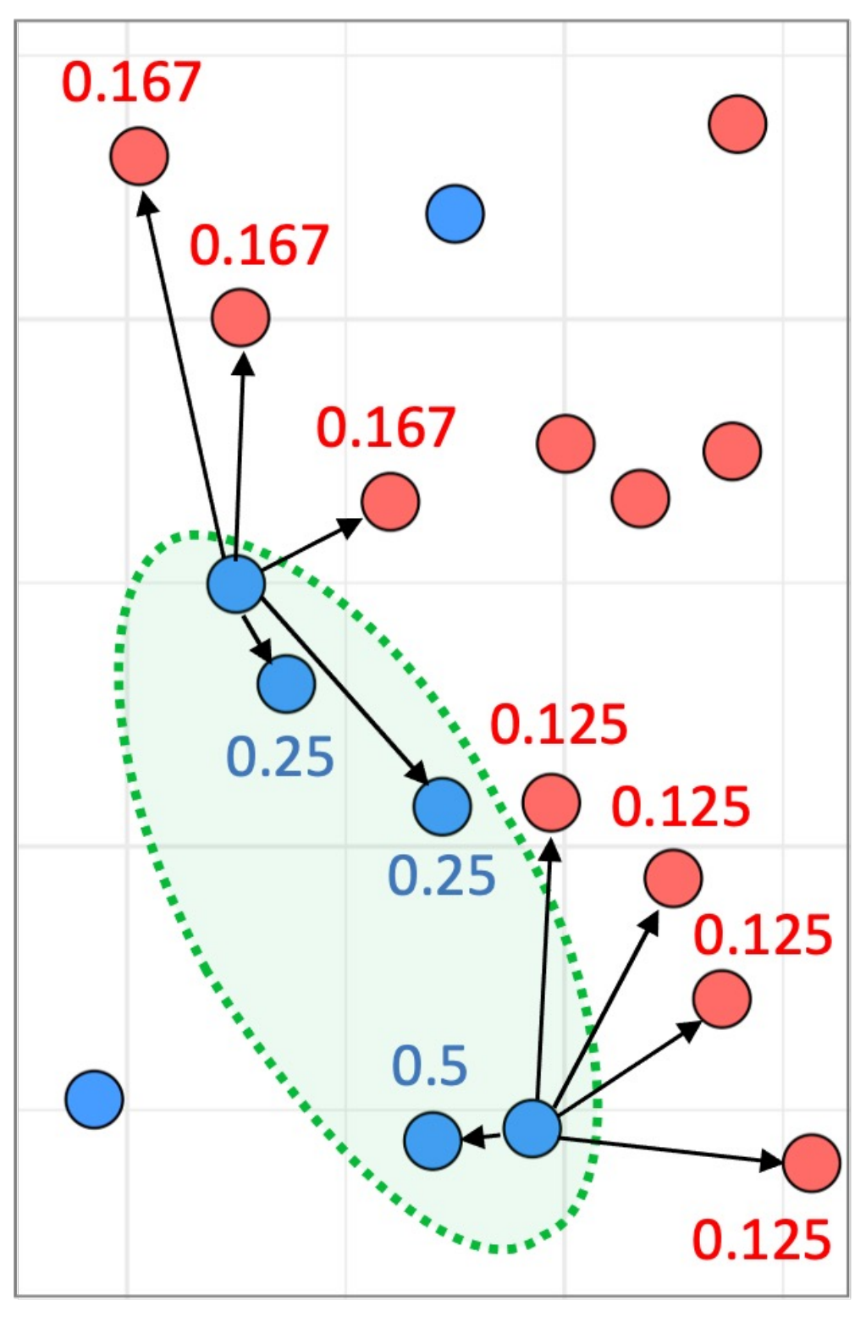
- Minority samples whose neighbors are all from the same class (minor) = *safe* minority samples
- Minority samples whose neighbors are all from the opposite class (major) = *noisy* minority samples
- The remaining minority samples are defined as *overlapping* minority samples



2. Synthetic Sample Generation

○ Generate $(n_1 - n_2)$ synthetic minority samples within the k-nearest neighbors of the overlapping minority samples

1) Randomly select an overlapping minority sample x_i
2) Compute neighbor weights w_{ij}
3) Proportionally select one neighbor x_{ij} with w_{ij}
   - a majority neighbors, $(k-a)$ minority neighbors
   - majority neighbor weight $= \frac{1}{2a}$
   - minority neighbor weight $= \frac{1}{2(k-a)}$
4) After selecting the neighbor x_{ij}, x_s is generated as



$$x_s = \lambda x_i + (1-\lambda) x_{ij}, \text{ where } \lambda \sim Unif(0,1) \quad \text{(numeric case)}$$

$$\text{or } x_s = \begin{cases} x_i, & with\ probability\ of\ \lambda \\ x_{ij}, & with\ probability\ of\ (1-\lambda) \end{cases} \quad \text{(discrete case)}$$

3. Filtering Synthetic Samples

○ Resample the previously generated synthetic minority samples by constructing sampling weights based on the first moment of the original minority samples

- For a continuous variable x_i, the sampling weights w_i are computed by maximizing the scaled Shannon entropy with a calibration constraint

$$w_i = \arg\max_w Q = \arg\max_w \left(-\sum_{i \in M_s} w_i \log w_i\right)$$

$$\text{s.t.} \begin{cases} \sum_{i \in M_s} w_i x_i = n_1^{-1} \sum_{j \in M_1} x_j \\ \text{where } \sum_{i \in M_s} w_i = 1, w_i > 0 \end{cases}$$

- For a discrete variable x_i, the sampling weights w_i are computed by applying a post-stratification method (Fuller, 2009)
- Using the obtained sampling weights w_i, proportionally select $(n_1 - n_2)$ synthetic minority samples with replacement

Experiment Results

Clf	Metric	SMOTE	ADASYN	BLSMOTE	SLMOTE	RSLSMOTE	BRnR
Rf	Gmean	0.705	**0.711**	0.634	0.569	0.569	0.69
Rf	F1	0.537	**0.539**	0.523	0.462	0.465	0.53
Rf	AUC	0.75	**0.754**	0.724	0.669	0.669	**0.754**
SVM	Gmean	0.738	0.736	0.684	0.637	0.635	**0.751**
SVM	F1	0.493	0.466	**0.512**	0.485	0.482	0.504
SVM	AUC	0.784	0.779	0.762	0.706	0.704	**0.789**
MLP	Gmean	0.774	0.769	0.692	0.646	0.643	**0.792**
MLP	F1	0.515	0.489	**0.528**	0.502	0.5	0.523
MLP	AUC	0.793	0.786	0.761	0.713	0.711	**0.805**
GBM	Gmean	0.768	0.777	0.696	0.64	0.642	**0.78**
GBM	F1	0.493	0.48	0.48	0.47	0.469	**0.502**
GBM	AUC	0.782	0.786	0.758	0.711	0.712	**0.791**

Table of mean score for each metric and classifier.

- Compared with 5 other oversampling methods by applying them to 20 datasets
- Methods: SMOTE, ADASYN, Borderline-SMOTE (BLSMOTE), Safe-level SMOTE (SLMOTE), Relocated Safe-level SMOTE (RSLSMOTE)
- Classifiers: Random Forest (Rf), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Gradient Boosting Machine (GBM)
- Metrics: G-mean, F1-score, AUC score
- Our method **outperforms other oversampling methods for most cases**
- Classifier-wise: obtained especially good results with **GBM**
- Metric-wise: obtained especially good results in **G-mean & AUC scores**

A Calibration-based Resampling Approach for Binary Imbalanced Classification [*]

Dongeun Min[†] Haehwan Lee[‡] Jongho Im[§]

Abstract

Imbalanced classification refers to classification of data where one class consists of a smaller number of samples than the other class. Class imbalance remains to be a challenging problem in machine learning. In this paper, we introduce a resampling method which constructs a balanced set by relabeling the data of each class, and generating synthetic samples for overlapping minority samples based on the class distribution of its nearest neighbors. When resampling each class, we use 2 main methods: raking and overlap detection. By generating synthetic minor samples with overlapping minority samples and resampling them with raking weights, we generate synthetic samples that help improve classification accuracy while suppressing them to be too far from the original sample mean. Through several experiments, we show that our method outperforms other sampling-based methods.

1 Introduction

Imbalanced classification refers to the classification of data where one class consists of a smaller number of samples than the other class (or classes). For the case of binary classification, the class which has more samples than the other is commonly defined as the "majority class", whereas the other class is defined as the "minority class". The class imbalance problem is found in various real-world domains, such as fraud detection [4] and bankruptcy prediction [3] where fraud or bankrupt instances are an extremely small fraction of the dataset.

Standard classification models assume balanced class distributions with equal misclassification costs for each class [1]. Thus, the classifier tends to sacrifice the classification accuracy of the minority class for the classification accuracy of the majority class. However, in many domains, the minority class is often the class that we are more interested in. Therefore, the main objective of imbalanced classification should be to accurately classify the minority instances while not missing majority instances as much as possible.

Methods for solving the class imbalance problem have been proposed from various viewpoints. Specifically, these methods can be categorized into 3 types: data-level methods, algorithm-level methods, and hybrid methods [2]. While data-level approaches attempt to balance class distributions by modifying the data, algorithm-level methods attempt to mitigate the classification bias of imbalanced data by directly adjusting the classifier. Methods such as cost-sensitive learning are part of algorithm-level approaches. Hybrid methods aim to attain the advantages of the data-level and algorithm-level methods by combining them.

Although there is no single approach that generally works well for all forms of imbalanced data, data-level sampling approaches have been shown to be useful in improving classification results [14], [7]. In this paper, we focus on data-level approaches in the problem setting of binary imbalanced classification. Data-level methods can be further separated into 3 types: oversampling, undersampling, and hybrid methods. Oversampling methods intend to make balanced data by generating new samples to the minority class, while undersampling methods construct balanced data by removing samples from the majority class.

Oversampling methods and undersampling methods have their own strengths and weaknesses. While oversampling methods construct balanced data by adding new synthetic samples, they are potentially vulnerable to overfitting [6]. Undersampling methods are free from this shortcoming, but these methods contain the danger of losing important feature information of the majority class. One of the most commonly used oversampling methods is SMOTE(Synthetic Minority Over-sampling Technique) [8]. SMOTE oversamples the data by randomly generating synthetic minority samples between the k-nearest neighbors of the minority samples. Many other oversampling methods that were developed after SMOTE are also the form of SMOTE variants, such as Borderline-SMOTE [10], ADASYN [9], and Safe-level SMOTE [11]. These SMOTE variants are distance-based methods in that they consider the class

[*]The full version of the paper can be accessed at `https://arxiv.org/abs/1902.09310`

[†]Yonsei University, Department of Statistics and Data Science

[‡]Yonsei University, Department of Statistics and Data Science

[§]Yonsei University, Department of Applied Statistics distribution of the k-nearest neighbors of each sample when selecting the area to generate additional samples.

RnR [20] differs from the methods above in that it uses the first moment of each class when selecting samples of each class to relabel and resample. However, RnR also has limitations in that it only generates additional minority samples by relabeling majority samples, not generating new synthetic samples. This makes the method sensitive to the number of how many samples will be relabeled.

In this paper, we propose a calibration-based resampling method which filters out noisy synthetic samples generated in the overlapping region of majority and minority samples. Our method generates synthetic minority samples in the detected overlapping minority region and removes noisy synthetic samples by resampling the minority samples with sampling weights constructed under a calibration constraint based on the first moment information. This procedure removes synthetic minority samples far from the first moment of the original minority set. Through extensive experiments comparing our method with commonly used oversampling methods, we show that our proposed approach obtains a significantly better classification performance in most cases.

The remainder of the paper is organized as the following: First, we review oversampling methods commonly used for imbalanced data. Here, we summarize the key ideas of some methods, including SMOTE variants and RnR. Second, we explain the details of our proposed algorithm. Third, we show the experimental setup and results of experiments we have implemented to compare the proposed method and other state-of-the-art oversampling methods. Lastly, we summarize our paper and propose future works that can be done.

2 Related Works

As mentioned previously, approaches for imbalanced classification generally fall into three categories: data-level methods, algorithm-level methods, and hybrid methods. Moreover, data-level methods can be divided again into three types: oversampling methods, undersampling methods, and hybrid methods. The proposed method of this paper can be classified as an oversampling method. Here, we review some commonly used oversampling methods that inspired our approach.

2.1 SMOTE Variants SMOTE [8] is one of the most commonly used oversampling methods for imbalanced data. SMOTE oversamples the minority data by generating synthetic minority samples in the line segment of two randomly selected minority samples. SMOTE generates additional minority samples in an intuitive way, but it has certain drawbacks that may affect classification accuracy. First, when some outliers exist, synthetic samples can be generated in a noisy area. Moreover, synthetic samples can also be generated in areas that are completely separated from majority samples and only consist minority samples, which does not contribute to finding the correct decision boundary. To overcome these drawbacks, several substitutes or variants of SMOTE have been introduced.

Borderline-SMOTE [10] is one of the methods that attempted to overcome these drawbacks. The key idea of Borderline-SMOTE is to generate synthetic samples within the borderline of majority and minority samples, since this area is tricky to accurately classify. Borderline-SMOTE finds borderline minority samples based on the nearest neighbors of each minority sample, and generates synthetic minority samples within these borderline samples.

Moreover, ADASYN [9] attempted to expand this idea by generating a different number of synthetic samples for each minority sample, based on the distribution of majority samples within its nearest neighbors. ADASYN assigns a weight to each minority sample depending on the number of majority samples that are in its nearest neighbors, and uses these weights in a form of a probability distribution that decides how many samples will be generated for each minority sample.

Furthermore, Safe-level SMOTE [11] aims to generate synthetic minority samples in a "safe area", where harmful or noisy synthetic samples are prevented from being generated. This method attempts to find the safe area to generate synthetic samples using the degree of minority samples in the nearest neighbors of each minority sample. Inspired from Safe-level SMOTE, Relocating Safe-level SMOTE(RSLS) [12] aims to improve Safe-level SMOTE by relocating the generated synthetic minority samples, so that they are far enough from possibly surrounding majority samples.

2.2 RnR The SMOTE variants above are distance-based methods in that they use the k-nearest neighbors of each sample when selecting the area to generate synthetic samples. Thus, these methods do not use the distribution information of each class. Unlike these methods, the RnR(Raking and Relabeling) [20] method generates a balanced set by relabeling part of the majority samples to minority samples and resampling each of the majority and minority samples. In other words, while other SMOTE variants make additional minority samples by generating synthetic minority samples, the RnR method brings additional minority samples from the majority samples which are already given. The key idea of RnR is to resample the data based on the sample means of the minority class and majority class. Through this process, both the majority samples and minority samples are resampled close to each of the sample means of the original majority class and minority class, which modifies the data of both classes to form a single cluster near the sample mean of each.

Theoretical backgrounds of why the RnR approach successfully resamples the data can be seen from the paper of Owen [21]. According to [21], when assuming an infinitely imbalanced set of samples and classifying them with logistic regression, the MLE(Maximum Log-likelihood Estimator) of the coefficient β of logistic regression remains unchanged regardless of using a centered log-likelihood or an uncentered log-likelihood.

That is, assuming an imbalanced data distribution, the coefficients of logistic regression are deeply related, and almost totally depend to the first moment of the samples. From this discovery, we can imply that removing outliers and redesigning the data to form a single cluster near the sample mean can help improve classification accuracy. Of course, we cannot conclude that this discovery applies to all classifiers. Therefore, through experiments with various datasets and classifiers other than logistic regression, the RnR method proves that this approach shows a good performance compared to other state-of-the-art methods, such as SMOTE, ADASYN, or MWMOTE [14].

However, the RnR method has two limitations. First, the RnR method assumes that the majority and minority class have the same support. That is, it is weak to the class disjunct problem [22]. Second, since the RnR method brings additional minority samples by relabeling the observed majority samples, it contains the danger of generating noisy minority samples. If too many samples are relabeled, the classification boundary will be greatly affected. Therefore, in this paper, we introduce a new resampling method inspired by RnR, added with distance-based overlap detection and synthetic sample generation. Through this method, we obtain a balanced set of data, where synthetic minor samples are generated in the overlapping area and noisy samples of each class are filtered out.

3 Proposed Method

Let M_0 and M_1 indicate the set of majority class samples and minority class samples respectively. The size of M_0 equals to n_0 and the size of M_1 equals to n_1. Since we assume the data is imbalanced, n_0 is larger than n_1. Moreover, $\mathbf{x}_i \in M_0$ and $\mathbf{x}_j \in M_1$ indicate the design variables of the majority and minority class. Our goal is to construct a balanced set of n_0 majority samples and n_0 minority samples.

Our proposed method consists of three steps. First, we start with detecting the overlapping minority samples. Second, we generate synthetic minority samples inside the overlapping minority samples and its nearest neighbors. Here, we select which neighbor to generate synthetic samples with based on the class distribution of the nearest neighbors. Finally, we resample the new minority set with raking weights and construct the final balanced set. The detailed process of the proposed algorithm is given above (Algorithm 1).

Algorithm 1 Proposed Method

```
Input: Initial samples $M = M_0 \cup M_1$
Output: Balanced samples $M^b = M_0^b \cup M_1^b$
  1. Overlapping area detection
  $M_{ov} = \varnothing$
  for $X_i \in M_1$ do
      Find $k$-nearest neighbors $X_{i1}, X_{i2}, \cdots, X_{ik}$
      if $(\exists j\ s.t.\ X_{ij} \in M_0)$ & $(\exists j\ s.t.\ X_{ij} \in M_1)$ then
          $M_{ov} = M_{ov} \cup \{X_i\}$
  2. Synthetic Sample Generation
  $M_s = \varnothing$
  for $(n_0 - n_1)$ times do
      1) Select $X_i \in M_{ov}$ s.t. $\{X_{ij}\}$ $(j = 1, 2, \cdots, k)$
         $= k$-nearest neighbors of $X_i$
      1) Compute neighbor weight $w_{ij}$ for $X_{ij}$
      2) Proportionally select 1 neighbor with $w_{ij}$
      3) Select $\lambda \sim Unif(0, 1)$
      4) Generate synthetic sample $X_s$
         $X_s = \lambda X_i + (1 - \lambda) X_{ij}$
      $M_s = M_s \cup \{X_s\}$
  3. Filtering Synthetic Samples
  1) Compute raking weights $\{w_i\}$ with original
     minority sample mean $\bar{X}_1$
  2) Proportionally select $(n_0 - n_1)$ samples from $M_s$
     with $\{w_i\}$
```

The proposed method is inspired by two key ideas: First, we generate synthetic minority samples within the k-nearest neighbors of the overlapping minority samples. When selecting the minority sample's neighbor to generate synthetic samples, we select a neighbor based on the neighbor weights. The neighbor weight equals to the probability to select each neighbor, which is calculated based on the ratio of majority and minority neighbors of the overlapping minority sample. Through selecting neighbors based on the distribution of majority and minority samples, it is possible to generate more synthetic samples on the borderline of the two classes, since samples that have both majority and minority neighbors are more likely to be the tricky borderline samples. Since minority samples whose neighbors are all from the opposite class are likely to be noisy samples,

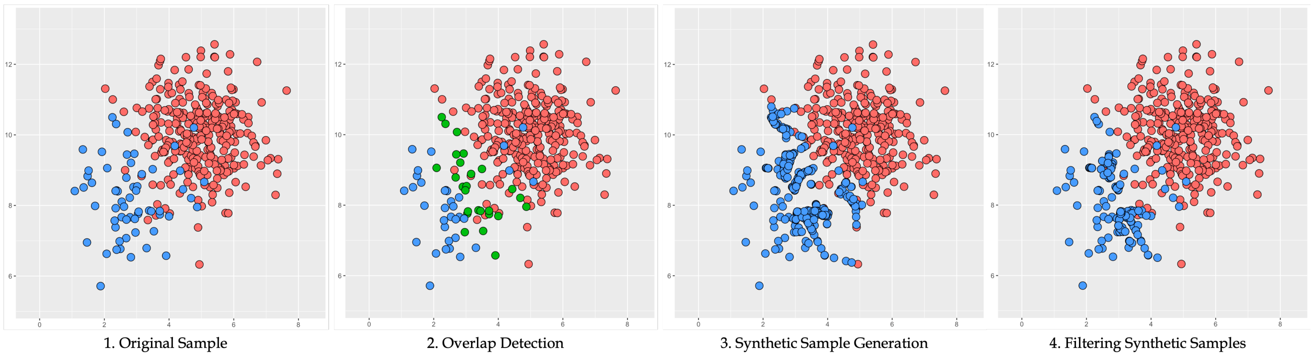


Figure 1: Process of the proposed method. From left, the plots show each step of our framework, where the first plot shows the original samples (red: major, blue: minor), the second plot shows the detected overlapping area (green), the third plot shows the generated synthetic samples, and the fourth plot shows the final result where synthetic samples are filtered.

these are excluded from generating synthetic samples. Minority samples whose neighbors are all from the same class are also excluded, since they are likely to be safe-area samples which are not tricky to classify, and thus do not need to be concentrated while generating additional synthetic samples.

Second, we resample the generated synthetic samples to filter out noisy samples that are far from the original minority sample mean. In this step, we resample the synthetic samples by constructing raking weights based on the first moment of the original minority samples. Through this process, the first moment of the new minority samples is maintained to be the same as the original minority sample mean.

3.1 Overlapping Area Detection First, we start with detecting the overlapping area of the majority and minority samples. This step is done by finding the k-nearest neighbors of each minority sample. The default value of k is 5.

We define the set of overlapping minority samples as M_{ov}. For each minority sample, we find its k-nearest neighbors based on Euclidean distance. If there exists at least one neighbor from each of the majority and minority class, the sample is defined as an overlapping minority sample. That is, we assume that minority samples whose neighbors are all from the same class are located in the ”safe” area. Also, we assume that minority samples whose neighbors are all from the opposite class are ”noisy” samples. Other than these safe samples and noisy samples, the remaining minority samples are defined as overlapping samples.

3.2 Synthetic Sample Generation Next, synthetic minority samples are generated within the nearest neighbors of the overlapping minority samples. In this step, we find the k-nearest neighbors of each overlapping minority sample, and generate synthetic samples within the original sample and one of its nearest neighbor. The process below is repeated so that $(n_0 - n_1)$ synthetic samples are generated.

When selecting which neighbor to generate synthetic samples with, we select with a probability weight given to each neighbor. This probability weight is calculated based on the class distribution of the k-nearest neighbors. Specifically, for those that have more majority neighbors than minority neighbors, a higher weight is given to the minority neighbors, and for those that have more minority neighbors, a higher weight is given to the majority neighbors. If there are a majority neighbors and b minority neighbors where $a > b$, the minority neighbors are given a weight a/b times larger than the weight given to majority neighbors. For the opposite case where $a < b$, the majority neighbors are given a weight b/a times larger than the weight given to minority neighbors.

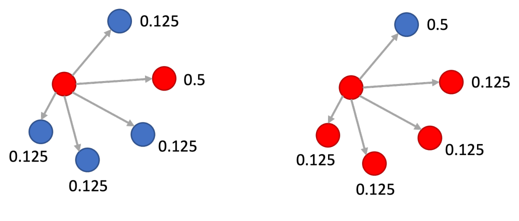


Figure 2: Neighbor weight of minority sample for $k = 5$

For example, let us assume that we select within the 5 nearest neighbors of an overlapping minority sample ($k = 5$). Then, if a selected overlapping minority sample has 4 majority neighbors and 1 minority neighbor, the minority neighbor is given a probability weight that is 4 times the weight of majority neighbors. Therefore, the minority neighbor is given a weight of 0.5 while the 4 majority neighbors are given a weight of 0.125 each. In contrast, if the selected overlapping minority sample has 1 majority neighbor and 4 minority neighbors, the majority neighbor is given a weight of 0.5, while the 4 minority neighbors are given a weight of 0.125 each.

Once we select which minority sample and neighbor to use, we generate a synthetic sample with these two samples. For numeric variables, the synthetic sample is calculated as the weighted sum of the minority sample and its neighbor, as in equation 3.1. Here, the weight λ is randomly selected from a Uniform distribution with a support of $[0, 1]$. For categorical variables, the synthetic sample is calculated by selecting either the minority sample variable or the neighbor variable with the probability of λ and $1 - \lambda$ respectively, as in equation 3.2. This process equals to mix up [23], which is commonly used for generating additional samples for image data in deep learning.

$$\mathbf{x}_s = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_{ij} \tag{3.1}$$

$$\mathbf{x}_s = \begin{cases} \mathbf{x}_i, & \text{with probability of } \lambda \\ \mathbf{x}_{ij}, & \text{with probability of } 1-\lambda \end{cases} \tag{3.2}$$

3.3 Filtering Synthetic Samples Finally, we resample the synthetic minority samples generated above through constructing sampling weights based on the first moment of the original minority samples. The goal here is to construct a final minority set of n_0 samples. The final minority set consists of n_1 original minority samples and $(n_0 - n_1)$ additional synthetic minority samples. The $(n_0 - n_1)$ synthetic samples are sampled with replacement from the synthetic minority samples M_s, generated in step **3.2**. Here, the final synthetic samples to be included are selected with probabilities of w_i, which are sampling weights constructed based on the original minority sample mean.

3.3.1 Continuous Variables For continuous variables, the sampling weights w_i are obtained via applying a calibration method [17], [18] on each variable $x_i^{(j)}$ $(j = 1, \cdots, q)$, where $x_i^{(j)}$ is the j-th element of $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(q)})$. Assuming that $x_i^{(j)}$ is a continuous variable, $w_i^{(j)}$ is computed by maximizing the scaled Shannon entropy with a calibration constraint as the following.

$$w_i^{(j)} = \arg\max_w Q = \arg\max_w \left(-\sum_{i\in M_s} w_i^{(j)} \log w_i^{(j)} \right) \tag{3.3}$$

$$\text{s.t.} \begin{cases} \sum_{i\in M_s} w_i \mathbf{x}_i = n_1^{-1} \sum_{j\in M_1} \mathbf{x}_j \\ \text{where } \sum_{i\in M_s} w_i^{(j)} = 1,\ w_i^{(j)} > 0 \end{cases}$$

Note that maximizing Q is equivalent to minimizing $-Q$. Therefore, the constrained maximization problem equals to a minimization problem as the following.

$$w_i^{(j)} = \arg\min_w \left(\sum_{i\in M_s} w_i^{(j)} \log w_i^{(j)} \right) \tag{3.4}$$

Using Lagrange multipliers λ, where λ is a 2×1 vector, the constrained optimization problem above can be transformed as minimizing the Lagrange function below.

$$\begin{aligned} &\arg\min_w \{ \sum_{i\in M_s} w_i^{(j)} \log w_i^{(j)} + \\ &\qquad (n_1^{-1} \sum_{i\in M_1} \lambda^T \mathbf{x}_i^{(j)} - \sum_{i\in M_s} w_i \lambda^T \mathbf{x}_i^{(j)})\} \end{aligned} \tag{3.5}$$

Here, $\mathbf{x}_i^{(j)} = (1, x^{(j)})$, where the constant term is combined to enforce the sum of $w_i^{(j)}$ to be equal to 1.

Note that $-Q$ is a non-negative, strictly convex function, differentiable with respect to $w_i^{(j)}$. Thus, $w_i^{(j)}$ can be expressed as an exponential function.

$$w_i^{(j)} \propto \exp\left(\lambda^T \boldsymbol{x}_i^{(j)}\right) \tag{3.6}$$

Therefore, by plugging in this exponential form, the calibration constraint of 3.3 can be rewritten as the following:

$$\sum_{i\in M_s} \exp\left(\lambda^T \mathbf{x}_i^{(j)}\right) \mathbf{x}_i^{(j)} - n_1^{-1} \sum_{j\in M_1} \mathbf{x}_j^{(j)} = 0 \tag{3.7}$$

Then, we can obtain the estimated sampling weights by solving the equation 3.7 with respect to λ with Newton-type algorithms.

3.3.2 Discrete Variables When $x_i^{(j)}$ is a discrete variable, we obtain the sampling weights $w_i^{(j)}$ by applying a post-stratification method [18]. Let $x^{(j)}$ be a discrete variable with K distinct values. Then, it is possible to split the synthetic sample set and minority sample set K subsets respectively.

$$M_{sk}^{(j)} = \{i \in M_s | x_i^{(j)} = k\},\ k = 1, \cdots, K \tag{3.8}$$
$$M_{1k}^{(j)} = \{i \in M_1 | x_i^{(j)} = k\},\ k = 1, \cdots, K$$

Then, the sampling weight for the j-th variable $w_i^{(j)}$ is computed as the following:

$$w_i^{(j)} = d_i \times \frac{n_{1k}^{(j)}/n_1}{\sum_{m \in M_{sk}^{(j)}} d_m},\ i \in M_{1k}^{(j)} \tag{3.9}$$
$$\text{where } n_{1k}^{(j)} = \text{size of } M_{1k}^{(j)}$$

By iteratively and sequentially repeating the weight computation process for each variable, we obtain the final sampling weights w_i. Using these weights as sampling probabilities, we proportionally select $(n_0 - n_1)$ synthetic samples with replacement, where those selected are included in the final minority sample set.

From this step, synthetic samples far from the original minority sample mean are removed, since those samples are given a low sampling weight and thus are not likely to be included in the final minority sample set. Therefore, this step can be seen as a synthetic sample filtering process, where noisy synthetic samples far from the original sample mean are filtered out.

4 Experiments

In order to evaluate the performance of our proposed method, we compared the classification performance of our method with five other oversampling methods widely used for imbalanced classification. Here, we apply each method to well-known datasets with various imbalanced ratio (IR) and size. In the section below, we describe our experimental setup. Then, we show the performance of our method together with five other oversampling methods.

4.1 Experimental Setup For comparison, we apply our method and 5 other commonly used oversampling methods to 20 datasets collected from the UCI machine learning repository [24] and KEEL(Knowledge Extraction based on Evolutionary Learning) repository [25]. Each of them are consisted of various numbers of samples (# S) and attributes (# Atts), along with different imbalance ratios(IR). A summary of the characteristics of each dataset is given in Table 1.

In the experiment, the following oversampling methods were applied for comparison: SMOTE [8], ADASYN [9], Borderline-SMOTE (BLS) [10], Safe-level SMOTE (SLS) [11], and Relocated Safe-level SMOTE (RSLS) [12]. Some approaches, such as Borderline-SMOTE, require their hyperparameters to be set as a certain value before implementation. For these approaches, the default value was given for each parameter. For example, for Borderline-SMOTE, k, which indicates the number of nearest neighbors to find, was set as $k = 5$. Also, since our proposed method transforms the data into a balanced set, other oversampling methods were also set to construct a balanced set, with the same number of majority samples and minority samples. Moreover, since methods such as SMOTE require all variables to be numeric, categorical variables were converted into dummy variables for a fair comparison.

When comparing results, we applied 5×2 fold cross-validation with 4 classifiers that are widely used recently: Random Forest(Rf), Support Vector Machine(SVM), Multi-layer Perceptron(MLP), and Gradient Boosting Machine(GBM) [30]. Again, for equal comparison, all parameters were set to the default value. The results of each method were measured with 3 different metrics: G-mean, F1-score, and AUC score. These metrics were chosen because all of them consider the classification accuracy of both the majority class and minority class. For example, the G-mean refers to the squared root of the product of sensitivity and specificity, the F1-score refers to the harmonic mean of precision and recall, and the AUC score refers to the area under the ROC curve.

Table 1: Description of Datasets

Dataset Name	IR	# Atts	# S
abalone-17_vs_7-8-9-10	39.31	8	2,338
abalone-19_vs_10-11-12-13	49.69	8	1,622
abalone-21_vs_8	40.5	8	581
abalone-9_vs_18	16.40	8	731
cargood	24.04	6	1,728
compustat	25.0	20	13,657
haberman	2.78	3	306
mammography	42.0	6	11,183
oil	22	49	937
pima	1.87	8	768
rice	2.32	8	3,810
vehicle0	3.25	18	846
vehicle3	2.99	18	846
winequality-red-4	29.17	11	1,599
yeast-0-2-5-6_vs_3-7-8-9	9.14	8	1,004
yeast-0-2-5-7-9_vs_3-6-8	9.14	8	1,004
yeast-0-5-6-7-9_vs_4	9.35	8	528
yeast3	8.1	8	1,484
yeast5	32.73	8	1,484
yeast6	41.4	8	1,484

Table 2: Mean Metric Rank of Each Method

Clf	Metric	SMOTE	ADASYN	BLSMOTE	SLMOTE	RSLSMOTE	Cal
Rf	Gmean	2.737	2.5	3.684	4.974	4.763	**2.342**
Rf	F1	3.421	**3.158**	3.184	4.263	3.763	3.211
Rf	AUC	2.895	2.684	3.605	4.947	4.526	**2.342**
SVM	Gmean	2.789	2.895	3.605	4.474	4.816	**2.421**
SVM	F1	3.711	4.263	3.289	**3.158**	3.421	**3.158**
SVM	AUC	2.684	3.105	3.553	4.395	4.684	**2.579**
MLP	Gmean	2.579	2.921	4.211	4.658	4.711	**1.921**
MLP	F1	3.658	4.342	3.526	**2.947**	3.184	3.342
MLP	AUC	2.632	3.026	4.132	4.553	4.763	**1.895**
GBM	Gmean	3.026	2.658	4.237	4.447	4.132	**2.5**
GBM	F1	3.553	3.947	4.105	**3.053**	3.237	3.105
GBM	AUC	2.974	2.737	4.158	4.368	4.158	**2.605**

Table 3: Mean Metric Score of Each Method

Clf	Metric	SMOTE	ADASYN	BLSMOTE	SLMOTE	RSLSMOTE	Cal
Rf	Gmean	0.705	**0.711**	0.634	0.569	0.569	0.69
Rf	F1	0.537	**0.539**	0.523	0.462	0.465	0.53
Rf	AUC	0.75	**0.754**	0.724	0.669	0.669	**0.754**
SVM	Gmean	0.738	0.736	0.684	0.637	0.635	**0.751**
SVM	F1	0.493	0.466	**0.512**	0.485	0.482	0.504
SVM	AUC	0.784	0.779	0.762	0.706	0.704	**0.789**
MLP	Gmean	0.774	0.769	0.692	0.646	0.643	**0.792**
MLP	F1	0.515	0.489	**0.528**	0.502	0.5	0.523
MLP	AUC	0.793	0.786	0.761	0.713	0.711	**0.805**
GBM	Gmean	0.768	0.777	0.696	0.64	0.642	**0.78**
GBM	F1	0.493	0.48	0.48	0.47	0.469	**0.502**
GBM	AUC	0.782	0.786	0.758	0.711	0.712	**0.791**

As a result, for 20 datasets, 4 classifiers, and 3 performance metrics, we calculated the average performance after 30 trials of each method. For implementation, the R packages **randomForest** [26], **e1071** [27], **imbalance** [28], and **smotefamily** [29] were used.

4.2 Results The experimental results on 20 datasets with 4 different classifiers are shown in Tables 2, 3, and 4. Table 2 shows the mean score of each oversampling method for each metric. Table 3 shows the mean rank of the compared methods for each metric. Since a total of 6 methods are compared, for each dataset, the 6 methods are ranked in descending order. Then, we calculate the mean rank for each metric and method. Lastly, table 4 shows the mean rank of each method calculated with all metrics. The best mean score or rank for each row is shown in bold.

From Tables 2 and 3, we can see that our method outperforms other state-of-the-art oversampling methods for most classifiers and metrics. In particular, our method achieves better performance, especially with G-mean and AUC scores. Although we do not obtain the best F1-scores for some classifiers, we still achieve the 2nd or 3rd best result. Methods that obtain a better F1-score than our method are mostly Borderline-SMOTE (BLSMOTE) and Safe Level SMOTE (SLMOTE). Compared to our method, these methods obtain much lower G-mean and AUC scores. Thus, we achieve the best mean rank for all classifiers, as we can see in Table 4. Furthermore, from Table 3, we can see that our method shows a particularly good performance for all metrics when classifying with gradient-boosting classifiers.

Clf	SMOTE	ADASYN	BLS	SLS	RSLS	Cal
Rf	3.02	2.78	3.49	4.73	4.35	**2.63**
SVM	3.06	3.42	3.48	4.01	4.31	**2.72**
MLP	2.96	3.43	3.96	4.05	4.22	**2.39**
GBM	3.18	3.11	4.17	3.96	3.84	**2.74**

Table 4: Mean Rank of Each Method

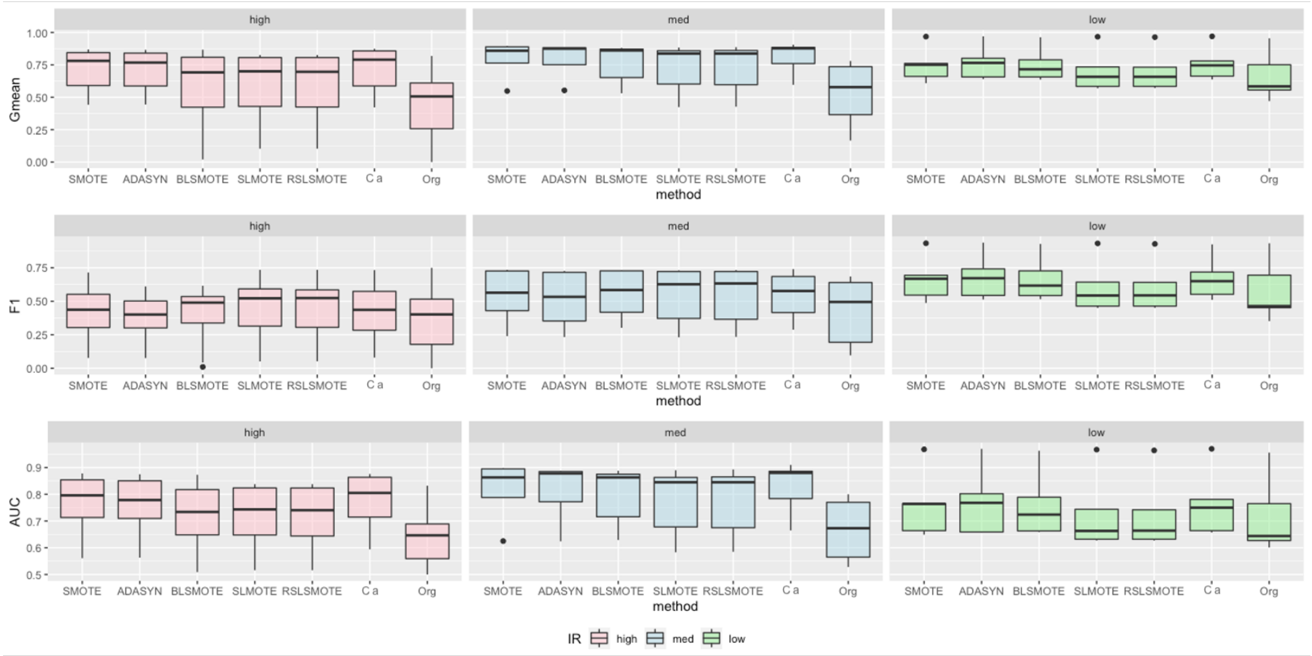


Figure 3: Comparison of G-mean, F1-score, AUC score for datasets with high/medium/low IR

Moreover, in order to see how the classification results differ by the dataset characteristics, we compared the G-mean, F1-score, and AUC scores of each method for datasets with high, medium, or low imbalance ratios (IR). Here, we define datasets with low imbalance ratios as those whose imbalance ratios are lower than 10. Datasets with imbalance ratios higher than 30 were defined as datasets with high imbalance ratios, and the remaining datasets were defined as those with medium imbalance ratios. The comparison box plot for each case is shown in Figure 3.

From Figure 3, we can see that for datasets with medium or low imbalance ratios, our method obtains a similarly good result for all metrics (G-mean, F1-score, and AUC score). For datasets with high imbalance ratios, our method obtains the best G-mean and AUC scores, but obtains a slightly lower F1-score compared to Borderline-SMOTE, Safe Level SMOTE, and Relocated Safe Level SMOTE. This is because for these datasets, our method obtains a higher recall but lower precision. From this, we can infer that for datasets with extremely rare minority samples, our method generates more synthetic samples in the borderline area than needed. This can happen because when minority samples are extremely rare, the first moment of the original minority samples may not be an accurate reference value, and thus the synthetic samples are not properly filtered in step **3.3**. However, in some class imbalance situations such as fraud detection, obtaining high recall is more important than obtaining high precision [31]. Since our method obtains the best G-mean and AUC scores, our method can still be useful for highly imbalanced cases.

5 Conclusion

In this paper, we propose a novel resampling method that successfully transforms imbalanced data into a balanced set so that better classification is made possible. Our method has strengths in two points. First, we successfully generate synthetic minority samples in the borderline area by considering the class distribution of the k-nearest neighbors of each minority sample. Borderline samples are often tricky to accurately classify, so these samples help improve the classification accuracy of the minority class. Moreover, our method filters noisy synthetic minority samples out by resampling the synthetic samples with sampling weights constructed based on the original minority sample mean. This resampling process ensures that the final minority sample set does not contain noisy samples far from the original minority sample mean.

From experiments with various datasets and classifiers, we show that our method brings a better classification result compared to other oversampling methods commonly used for imbalanced data. Further work can be done for showing whether our method also outperforms state-of-the-art methods for noisy imbalanced classification, where class noise exists with a certain probability for both classes.

References

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

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

[3] D. Veganzones and E. Séverin, *An investigation of bankruptcy prediction in imbalanced datasets*, Decision Support Systems, 112 (2018), pp. 111-124.

[4] S. Makki, Z. Assaghir, Y. Taher, R. Haque, M. S. Hacid, and H. Zeineddine, *An experimental study with imbalanced classification approaches for credit card fraud detection*, IEEE Access, 7 (2019), 93010-93022.

[5] Y. Sun, A. K. C. Wong, and M. S. Kamel, *Classification of imbalanced data: A review*, International journal of pattern recognition and artificial intelligence 23, 04 (2009), pp. 687-719.

[6] N. V. Chawla, *Data mining for imbalanced datasets: An overview*, Data mining and knowledge discovery handbook, (2009), 875-886.

[7] I. Nekooeimehr and S. K. Lai-Yeun, *Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets*, Expert Systems with Applications, 46 (2016), pp. 405-416

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *SMOTE: synthetic minority over-sampling technique*, Journal of artificial intelligence research, 16 (2002), pp. 321-357.

[9] H. He, Y. Bai, E. A. Garcia, and S. Li, *ADASYN: Adaptive synthetic sampling approach for imbalanced learning*, IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), 2008, pp. 1322-1328.

[10] H. Han, W. Y. Wang, and B. H. Mao, *Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning*, International conference on intelligent computing, Springer, Berlin, Heidelberg, 2005, pp. 878-887.

[11] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, *Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem*, Pacific-Asia conference on knowledge discovery and data mining, Springer, Berlin, Heidelberg, 2009, pp. 475-482.

[12] W. Siriseriwan and K. Sinapiromsaran, *The effective redistribution for imbalance dataset: relocating safe-level SMOTE with minority outcast handling*, Chiang Mai J. Sci 43, 1 (2016), pp. 234-246.

[13] G. Douzas and F. Bacao, *Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE*, Information Sciences, 501 (2019), pp. 118-135.

[14] S. Barua, M. M. Islam, X. Yao, and K. Murase, *MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning*, IEEE Transactions on knowledge and data engineering, 26.2 (2012), pp. 405-425.

[15] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, *DBSMOTE: density-based synthetic minority over-sampling technique*, Applied Intelligence, 36(3), (2012), pp. 664-684.

[16] M. Ester, H. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, In Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), (1996), pp. 226–231.

[17] J. C. Deville, C. E. Särndal, and O. Sautory, *Generalized raking procedures in survey sampling*, Journal of the American statistical Association 88, 423 (1993), pp. 1013-1020.

[18] W. A. Fuller, Sampling Statistics, Wiley & Sons, Inc, (2009).

[19] J. Im, Y. Seo, K. S. Cetin, and J. Singh, *Energy efficiency in us residential rental housing: Adoption rates and impact on rent*, Applied Energy, vol. 205 (2017), pp. 1021–1033.

[20] S. Park, H. H. Lee, and J. Im, *Raking and relabeling for imbalanced data*, (2022).

[21] A. B. Owen, *Infinitely imbalanced logistic regression*, Journal of Machine Learning Research, vol. 8 (2007), pp. 761–773.

[22] T. Jo and N. Japkowicz, *Class imbalances versus small disjuncts*, ACM Sigkdd Explorations Newsletter 6, no. 1 (2004), pp. 40-49.

[23] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, *mixup: Beyond empirical risk minimization*, (2017), arXiv preprint arXiv:1710.09412.

[24] D. Dua and C. Graff, *UCI machine learning repository*, Irvine, CA: University of California, School of Information and Computer Science, 2019, [http://archive.ics.uci.edu/ml].

[25] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, *Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework*, 2011, [https://sci2s.ugr. es/keel/datasets.php].

[26] L. Breiman, A. Cutler, A. Liaw, and M. Wiener, *Breiman and Cutler's Random Forests for Classification and Regression*, 2018, [https://cran.r-project.org/web/packages/randomForest/randomForest.pdf].

[27] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.-C. Chang, and C.-C. Lin, Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien, 2021, [https://cran.r-project.org/web/packages/e1071/e1071.pdf].

[28] I. Cordón, S. García, A. Fernández, and F.Herrera, *Imbalance: Oversampling algorithms for imbalanced classification in R*, Knowledge-Based Systems, 161 (2018), pp.329-341, [https://doi.org/10.1016/j.knosys.2018.07.035].

[29] W. Siriseriwan, *smotefamily: a collection of oversampling techniques for class imbalance problem based on SMOTE*, R Package Version 1.3.1, no. 1 (2019), [https://cran.r-

project.org/web/packages/smotefamily/smotefamily.pdf].

[30] J.H. Friedman, *Greedy function approximation: a gradient boosting machine*, Ann. Stat. 29 (5) (2001), pp. 1189–1232

[31] S. Wang and X. Yao, *Diversity analysis on imbalanced data sets by using ensemble models*, IEEE symposium on computational intelligence and data mining, (2009), pp. 324-331, IEEE.