MixBoost: Synthetic Oversampling using Boosted Mixup for Handling Extreme Imbalance

Anonymous Author(s)*

ABSTRACT

Training a classification model on a dataset where the instances of one class outnumber those of the other class is a challenging problem. Such imbalanced datasets are standard in real-world situations such as fraud detection, medical diagnosis, and customer churn prediction. We propose a data augmentation method, *MixBoost*, which intelligently selects (*Boost*) and then combines (*Mix*) instances from the majority and minority classes to generate synthetic hybrid instances that have elements of both classes. We evaluate *MixBoost* on 20 benchmark datasets and show that it outperforms existing approaches. We evaluate the impact of the different components of *MixBoost* using ablation studies. All of our code is available at https://github.com/user1332/MIXBOOST-code.

KEYWORDS

imbalanced datasets, neural networks, sampling, minority sampling, data augmentation, class imbalance

ACM Reference Format:
Anonymous Author(s). 2018. MixBoost: Synthetic Oversampling using Boosted Mixup for Handling Extreme Imbalance. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Several real-world situations (fault detection [29], disease classification [20], software failures [4], customer churn prediction [5], oil spill detection [18] and protein sequence detection [2]) involve learning from datasets where the instances of one class (the majority class) far outnumber those of the other class (the minority class). Training a binary (two-class) supervised classification model on such imbalanced datasets is a challenging problem.

A popular class of methods alleviates this problem by augmenting the training data with synthetic instances before training the classification model [6, 7, 14, 22]. These data augmentation methods generate synthetic minority class instances using either minority or majority class instances from the training dataset. However, existing methods often generate instances that do not improve (or even worsen) classifier performance. Intuitively, these methods generate good quality synthetic instances in regions of the input space where the classification model is already accurate. In regions where the model is inaccurate and requires synthetic instances, these methods often do not perform as well. Figure 1 illustrates this intuition. We believe that using the trained model (whose performance we are trying to improve) to guide the data augmentation process will allow us to augment regions of the input space where the model performs poorly. Further, existing methods generate synthetic **homogeneous** instances, i.e., instances that belong to a single class (usually the minority class). Recent work in the domain of Computer Vision [23, 26, 30] has demonstrated the value of augmenting training datasets with non-homogeneous **hybrid** instances to learn more robust representations.

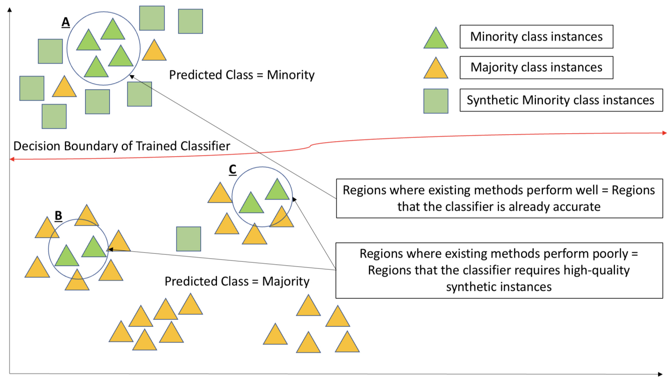

A
B
C
Predicted Class = Minority
Predicted Class = Majority
Decision Boundary of Trained Classifier
Minority class instances
Majority class instances
Synthetic Minority class instances
Regions where existing methods perform well = Regions that the classifier is already accurate
Regions where existing methods perform poorly = Regions that the classifier requires high-quality synthetic instances

Figure 1: Illustration to describe the limitation in existing data augmentation methods. Existing methods (SMOTE [6] and its variants, SWIM [22]) select instances at random and create synthetic instances based on the selected instances. Therefore, most synthetic instances lie near clusters of instances that are often already correctly classified by the classification model (region A). On the other hand, these methods generate fewer and poorer quality synthetic instances in regions of the input space where the model does not perform well (regions B and C).

We describe a data augmentation method (*MixBoost*) that improves classifier performance on such imbalanced datasets. Our key contributions are:

- We introduce a data augmentation method, *MixBoost* that generates synthetic **hybrid** instances to augment an imbalanced dataset (Section 2). *MixBoost* has two innovative components. **First**, it mixes instances of the minority and majority classes to generate synthetic hybrid instances that have elements of both classes (Section 2.4). **Second**, it intelligently selects the instances for mixing using a novel entropy-weighted **low-high** technique (Section 2.3).
- We show that *MixBoost* outperforms state-of-the-art data augmentation methods on highly imbalanced benchmark datasets for different levels of class imbalance (Section 4).

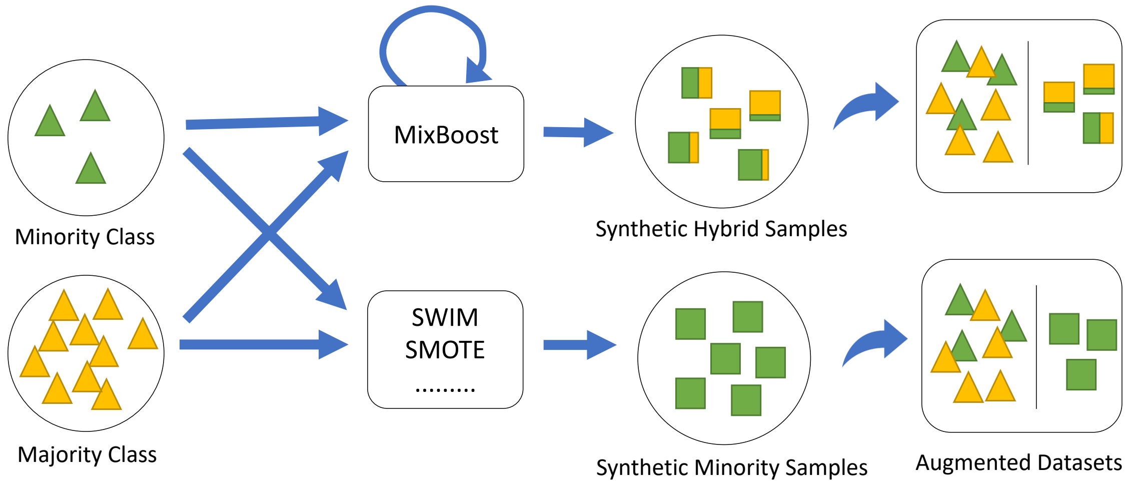


Figure 2: High-level summary of the data-augmentation pipeline. The triangle shape represents the data instances in the non-augmented training dataset. Green is for minority class and yellow for majority class. The square shape represents the synthetic generated instances that are added to the training dataset (augmentation) prior to training. Traditional approaches such as SWIM [22], SMOTE [6], and its variants generate synthetic minority class instances (denoted by green). In contrast, *MixBoost* generates synthetic hybrid instances that interpolate the minority and majority class instances (denoted by green-yellow color scheme).

There are several categories of methods to augment an imbalanced dataset prior to training a classification model. Under-sampling methods discard instances of the majority class at random to balance the class distribution. However, removing instances of the majority class can lead to a loss of information and subsequently degrade classifier performance. Over-sampling methods duplicate instances of the minority class at random to balance the class distribution. These methods add duplicates of existing instances to the training dataset and therefore do not add new information [12, 13]. More sophisticated data augmentation methods augment the training datasets by creating synthetic minority class instances. For instance, SMOTE [6] creates synthetic minority instances by interpolating minority class instances in the training data. However, SMOTE ignores majority class data when creating synthetic instances. Variants of SMOTE (Borderline SMOTE [15], ADASYN [17], SMOTEBoost [7]) use the distribution of majority class data to filter instances created using minority class data. However, since these methods use only minority class data, the synthetic instances they create are restricted to the convex-hull of the minority class distribution. To expand the space spanned by generated instances, SWIM [22] creates instances by inflating minority class data along the density contours of majority class data.

Existing data augmentation methods give promising results in a variety of situations. However, these methods focus on either the majority class or the minority class. Approaches that focus on the minority class often overlook the majority class and can degrade classifier performance by generating borderline or overlapping instances. On the other hand, approaches that focus on the majority class have a tendency to overfit the training distribution.

In contrast, *MixBoost* combines instances from the majority and minority classes to create synthetic hybrid instances. Each synthetic hybrid instance contains elements of a majority class and a minority class instance. Further, *MixBoost* uses an information-theoretic measure along with the trained classifier to select the instances to combine. Figure 2 illustrates the essential idea and distinction to related approaches.

Figure 3 shows the high-level architecture of *MixBoost*. *MixBoost* generates synthetic hybrid instances iteratively. Each iteration consists of a *Boost* step followed by a *Mix* step. In the *Boost* step, *MixBoost* uses the trained classifier to intelligently select pairs of instances from the minority and majority class for mixing. Intuitively, *MixBoost* selects pairs of instances such that one instance in each pair is close to the decision boundary of the trained classifier, and the other instance is far from the decision boundary. In the *Mix* step, *MixBoost* mixes the selected instances to create synthetic hybrid instances that are used to augment the training dataset. The classifier is re-trained with the original data augmented with the hybrid instances prior to the next iteration of *MixBoost*.

We evaluate *MixBoost* against existing state-of-the-art synthetic oversampling approaches, on 20 benchmark datasets (Section 4). In Section 4.2, we present ablation studies to evaluate the impact of different components of *MixBoost*.

2 APPROACH

Here we explain in detail the various steps of our proposed *MixBoost* algorithm. In Section 2.1 we define the terminologies used throughout the paper. In Section 2.2 we provide a high-level overview of the proposed approach followed by detailed explanation of the different stages in Sections 2.3 and 2.4.

2.1 Definitions

A binary classification task is characterized by a dataset $X^{n \times m} \in R$ and corresponding labels $Y^n \in 0, 1$. The class label 0 corresponds to

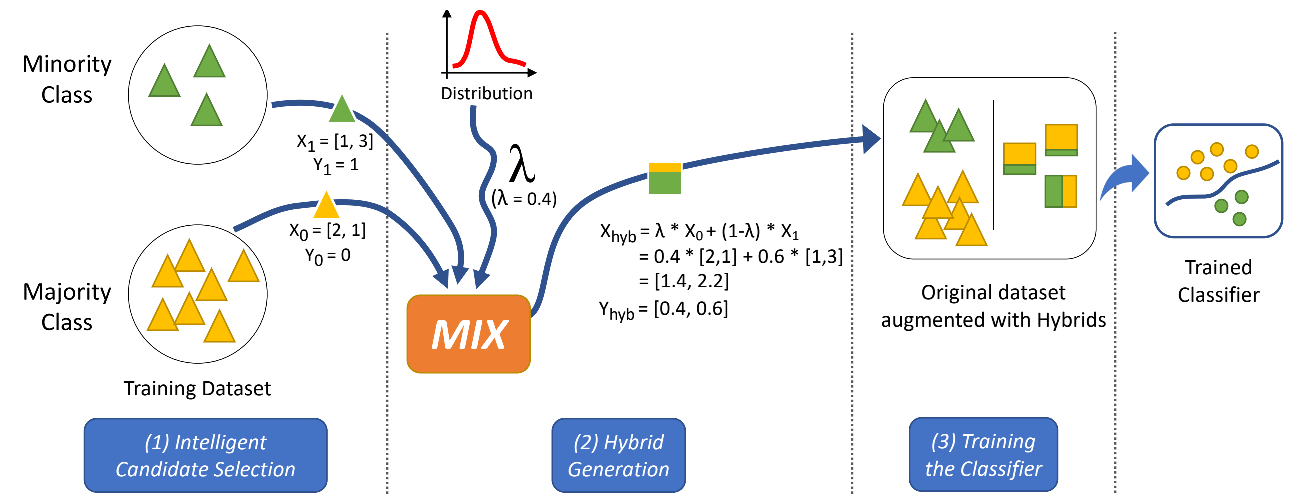


Figure 3: High-level architecture of *MixBoost*. In the (1) candidate selection step (*Boost*), *MixBoost* uses entropy to intelligently select instances from the training dataset. In the (2) hybrid generation step (*Mix*), we mix the selected instances to create synthetic hybrid instances. λ is the interpolation ratio that controls the extent of overlap between the instances to be mixed. We then augment the training dataset with the generated hybrid instances and train a classifier on the augmented dataset.

the majority class, and the class label 1 corresponds to the minority class throughout the paper. Let n be the number of instances in the dataset, and m be the number of features for an instance in the dataset. Then, the training objective is to learn a function, $f(x_i) \rightarrow y_i$, that maps the instances $x_i \in X$ to their corresponding labels $y_i \in Y$. For imbalanced datasets, the number of instances of the majority class far outnumbers that of the minority class. The dataset, denoted by (X, Y), is split into a training dataset (X_{train}, Y_{train}), and a testing dataset (X_{test}, Y_{test}). Let n_{train} be the number of instances in the training dataset. We train a binary classification model M on (X_{train}, Y_{train}). M is evaluated on the test dataset (X_{test}, Y_{test}). We evaluate the classifier M using the *g-mean* [19] and ROC-AUC score.

MixBoost uses the classifier M along with (X_{train}, Y_{train}) to create synthetic hybrid instances. The instances are added to the training dataset to create the augmented dataset $(X^{aug}_{train}, Y^{aug}_{train})$. The classifier M is then trained using the augmented dataset $(X^{aug}_{train}, Y^{aug}_{train})$. This represents a single iteration of *MixBoost*. *MixBoost* is run for k iterations to expand the augmented dataset. We compute all evaluation scores using the model trained on the final augmented dataset. All evaluation scores are computed on (X_{test}, Y_{test}).

2.2 MixBoost: High-Level Summary

MixBoost generates synthetic hybrid instances by interpolating instances from the majority and minority classes. A single iteration of *MixBoost* consists of the following steps:

(1) **Instance Sampling (*Boost*):** We sample instances from the majority and minority classes in (X_{train}, Y_{train}) prior to mixing. We experiment with both random and guided sampling techniques (Section 2.3).
(2) **Hybrid Instance Generation (*Mix*):** We mix the sampled instances to generate synthetic hybrid instances (Section 2.4). (X^{hyb}, Y^{hyb}) denotes the synthetic instances generated in this step.

At the end of a *MixBoost* iteration, we add (X^{hyb}, Y^{hyb}) to $(X^{aug}_{train}, Y^{aug}_{train})$. We then retrain the classifier M on $(X^{aug}_{train}, Y^{aug}_{train})$ before the next *MixBoost* iteration.

For the explanation that follows, we consider the problem of generating a single synthetic hybrid instance. The steps to generate several instances follows from the single instance case. *MixBoost* creates a synthetic hybrid instance by mixing a majority class instance (x_0) and a minority class instance (x_1). There are two important considerations. First, how should *MixBoost* select x_0 and x_1 from (X_{train}, Y_{train})? Second, how should *MixBoost* intelligently combine x_0 and x_1 to create the synthetic hybrid instance (x^{hyb}, y^{hyb})? We first consider the problem of instance selection.

2.3 MixBoost: Instance Selection (the *Boost* step)

We consider two alternative methods for instance selection.

2.3.1 *R-Selection*.

We select x_0 and x_1 from (X_{train}, Y_{train}) using Uniform Random Selection or *R-Selection* [30]. The probability of selecting an instance is constant across all instances of the majority and minority classes. Formally, if p^i_0 is the probability of selecting the i^{th} majority class instance and p^j_1 is the probability of selecting the j^{th} minority class instance, then:

$$p^i_0 = 1/n_0 \ \forall i \tag{1}$$

where n_0 is the number of majority class instances in (X_{train}, Y_{train}).

$$p^j_1 = 1/n_1 \ \forall j \tag{2}$$

where n_1 is the number of minority class instances in (X_{train}, Y_{train}).

2.3.2 *Entropy Weighted (EW) Selection*.

R-Selection assumes that mixing any two instances to create a hybrid instance is equally useful. However, it is possible that mixing certain instances in the training dataset is more useful than mixing others. Further, the instances selected for mixing should be a function of the trained classification model so that the generated synthetic instances improve the decision boundary of the model. We propose an Entropy-Weighted selection method (*EW-Selection*). *EW-Selection* is based on the intuition that mixing instances closer to the decision boundary of the trained classification model will generate more useful synthetic hybrid instances than mixing instances at random. Formally, if $\vec{y}^{i}_{pred}$ is the probability distribution over target classes output by the classifier M for the i^{th} instance in (X_{train}, Y_{train}), then the entropy of the sample, denoted by E^i, is computed as,

$$E^i = E(\vec{y}^{i}_{pred}) = -\sum y^{i}_{pred} * log(y^{i}_{pred}) \tag{3}$$

where the summation is over the elements of the vector $\vec{y}^{i}_{pred}$.

E^i measures the distance of the instance to the decision boundary of the classifier. A high E^i implies that M is uncertain about the ground-truth class for the instance. This case represents an instance that is close to the decision boundary of the classifier. Similarly, a low E^i implies that M is more certain about the ground-truth class for the instance. This case represents an instance that is far from the decision boundary of the classifier. Let $E_0 = \sum E^i \ \forall x_0$ is the sum of entropy values for majority class instances and $E_1 = \sum E^i \ \forall x_1$ denote the sum of entropy values for minority class instances. Then, p^i_0 denotes the probability of selecting a majority class instance for mixing, and p^j_1 denotes the probability of selecting a minority class instance for mixing. Both p^i_0 and p^j_1 are computed as,

$$p^i_0 = E^i / E^0, \qquad p^j_1 = E^j / E^1 \tag{4}$$

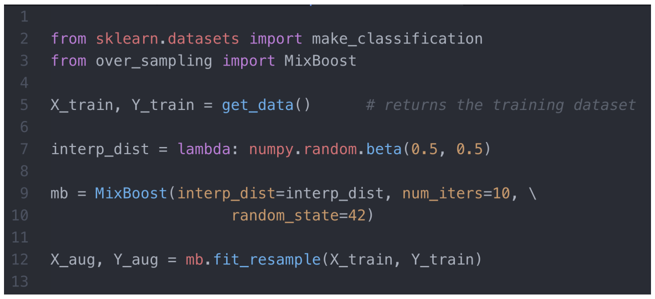
In practice, we find that selecting one candidate instance close to the decision boundary (high entropy) and one far from the boundary (low entropy) leads to synthetic hybrid instances that result in the best performance. *MixBoost* interleaves this **low-high** selection with the generation of hybrid instances using the *Mix* step described below.

2.4 MixBoost: Hybrid Generation (the *Mix* step)

Let (x_0, y_0) and (x_1, y_1) be instances selected (in the *Boost* step) from the majority and minority class respectively. We adopt the mixing strategy introduced in Zhang et al. [30].

$$\begin{aligned} x_{hyb} &= \lambda * x_0 + (1-\lambda) * x_1 \\ y_{hyb} &= \lambda * y_0 + (1-\lambda) * y_1 \end{aligned} \tag{5}$$

(x_{hyb}, y_{hyb}) is the generated synthetic hybrid instance. λ is the interpolation ratio. Intuitively, λ controls the extent of overlap between the two instances used to generate the synthetic hybrid instance. MIXUP [30] and BC+ [27] are existing image augmentation techniques which use linear and non-linear interpolations respectively. BC+ samples λ using the Uniform Distribution $U(0, 1)$ while MIXUP samples λ using the $Beta(\alpha, \alpha)$ distribution. For most tasks, MIXUP uses $\alpha = 1$, which also reduces to sampling λ from a Uniform Distribution [24]. In contrast, *MixBoost* works with skewed data distributions arising from extreme class imbalance. Correspondingly, we explore sampling λ from linear and non-linear probability density functions and discuss our findings in Section 4.2.3.



```

from sklearn.datasets import make_classification
from over_sampling import MixBoost

X_train, Y_train = get_data()      # returns the training dataset

interp_dist = lambda: numpy.random.beta(0.5, 0.5)

mb = MixBoost(interp_dist=interp_dist, num_iters=10, \
                        random_state=42)

X_aug, Y_aug = mb.fit_resample(X_train, Y_train)

```

Figure 4: Code snippet that illustrates how to use *MixBoost* in Python. All of our code along with examples are available here [8].

The resultant synthetic hybrid instances are used to augment the training dataset and (re)train the classifier. In the subsequent iteration, this classifier is used to update entropy values for candidates in the original training dataset to facilitate the next iteration of *MixBoost*. This workflow is shown in Figure 3.

3 EXPERIMENTAL SETTINGS

3.0.1 Datasets. To evaluate *MixBoost* against the state-of-the-art oversampling methods, we compare performance on binary classification on **20 benchmark datasets** (Table 1, available here[1]). To ensure evaluation consistency, we use the same datasets as used by Sharma et al. [22]. For robust evaluation, we split the dataset into equal training and testing halves. We randomly down-sample the minority class instances in the training dataset to simulate different levels of extreme imbalance. Specifically, we test at three levels of imbalance, with minority training dataset sizes of 4, 7, and 10. We further skew the data distribution (2 & 3 minority class instances) to show that *MixBoost* outperforms existing methods when using a fewer number of minority class training instances.

3.0.2 Classification. For all experiments, we use the Naive Bayes, Nearest Neighbour, Decision Tree, Support Vector Machine, and Multi-Layer Perceptron (MLP) binary classifiers. For each dataset, we compare the best performance of each data augmentation method across the various binary classifiers against the performance of *MixBoost*. Since *MixBoost* generates hybrid instances, we use only the MLP classifier to generate evaluation scores for *MixBoost*. We use the *sklearn* [21] package in Python for our experiments; all classifiers use the default parameters. We use *tensorflow* to implement the MLP. For the MLP, we use *Adam* optimizer, a learning rate of $1e-4$, a batch size of 500, and train it for 300 epochs. We use the same hyper-parameters for all datasets. We normalize the data before training and testing.

3.0.3 Data Augmentation. We use each data augmentation method to generate n synthetic instances, where n is the number of instances in the training dataset. For *MixBoost*, we find that generating the n instances over 5 iterations ($0.2n$ in each iteration) leads to the best results. We sample the interpolation ratio λ from a $Beta(0.5, 0.5)$ distribution. We explore the sensitivity of *MixBoost* to these hyper-parameters in Section 4.2.

[1]Datasets can be downloaded from Dua and Graff [9]

Dataset	Features	No. of majority instances	R4	R7	R10
Abalone 9-18	8	689	1:173	1:99	1:69
Diabetes	8	500	1:125	1:72	1:50
Wisconsin	9	444	1:111	1:64	1:456
Wine Q. Red 4	11	1546	1:387	1:221	1:155
Wine Q. White	11	880	1:220	1:126	1:88
Vowel 10	13	898	1:225	1:129	1:90
Pima Indians	8	500	1:125	1:72	1:50
Vehicle 0	18	641	1:160	1:91	1:64
Vehicle 1	18	624	1:156	1:89	1:62
Vehicle 2	18	622	1:155	1:88	1:62
Vehicle 3	18	627	1:156	1:89	1:62
Ring Norm	20	3736	1:934	1:534	1:374
Waveform	21	600	1:150	1:86	1:60
PC4	37	1280	1:320	1:183	1:128
Piechart	37	644	1:161	1:92	1:65
Pizza Cutter	37	609	1:153	1:87	1:61
Ada Agnostic	48	3430	1:858	1:490	1:343
Forest Cover	54	2970	1:743	1:425	1:297
Spam Base	57	2788	1:697	1:399	1:279
Mfeat Karhu.	64	1800	1:450	1:258	1:180

Table 1: Description of the datasets used in our experiments. To ensure evaluation consistency, we use the same datasets and configuration as proposed by Sharma et al. [22]. R4, R7, and R10 denote the ratio of class imbalance (minority:majority) after down-sampling the training datasets to have 4, 7, and 10 minority class instances respectively.

3.0.4 Evaluation. We compare data augmentation methods on two metrics. First, we use *g-mean* [19], which is the geometric mean of the True Positive Rate (TPR) (for majority class) and the True Negative Rate (TNR) (for minority class).

$$g\text{-}mean = \sqrt{TPR \times TNR} \tag{6}$$

We use *g-mean* because it is both immune to imbalance [19] and provides information about extreme imbalance [22]. Second, for completeness, we also compare methods using the ROC-AUC scores. We repeat all experiments 30 times and report the mean and standard deviation (rounded to the nearest decimal place) of the metrics for each dataset.

We compare *MixBoost* to Random Over-sampling and Undersampling (ROS, RUS), SMOTE [6], Borderline-SMOTE (B1, B2) [14], SMOTE with Tomek Links [19], ADASYN [17] and SWIM [22]. To make reporting easier, we merge all re-sampling approaches (ROS, RUS, SMOTE, B1, B2, SMOTE with Tomek Links, ADASYN) as Alternative Re-sampling Techniques (ALT) and report the best performing combination of data augmentation method and classifier for each dataset. We report the results from SWIM [22] separately since it is the most recent work, and it outperforms methods in ALT on most datasets. Finally, *Baseline* represents the case where no data augmentation is used before training the binary classifier.

4 RESULTS

Tables 2 and 3 show the results for our experiments on the *g-mean* and ROC-AUC metrics respectively. Though we present results for the ROC-AUC metric as well, *g-mean* is considered a better metric for evaluating performance when dealing with an extreme imbalance in datasets [19, 22].

Table 2 shows the results corresponding to the case where we down-sample the training dataset to have 4 minority class instances. We see that both *R-Selection* and *EW-Selection* (the two alternative selection strategies in the *Boost* step), outperform existing methods on **18 out of the 20** benchmark datasets. Further, *EW-Selection* is significantly better than *R-Selection* on several datasets. This difference between selection strategies for *MixBoost* indicates that selecting instances using entropy leads to the creation of more useful synthetic hybrid instances than selecting instances at random. For completeness, we compute the ROC-AUC scores for classifiers trained using *MixBoost* and SWIM. For *MixBoost*, we use *EW-Selection* for each dataset.

Table 3 shows the results on the ROC-AUC metric for all the datasets. We see that *MixBoost* outperforms SWIM on **19 out of the 20**.

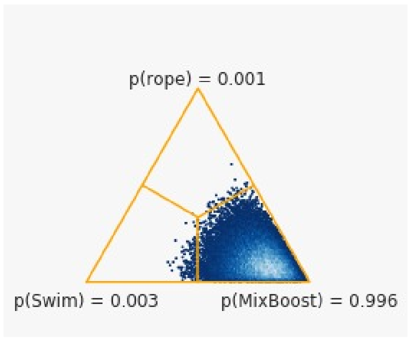
4.1 Statistical Significance

We use the Bayesian signed test [3] to evaluate the results presented in the previous section. The Bayesian signed test is used to compare two classification methods over several datasets. We use the test to compare *MixBoost* to SWIM over all 20 datasets. We compare the methods for training datasets down-sampled to 4, 7, and 10 minority class instances. Using the Bayesian method enables us to ask questions about posterior probabilities, that we cannot answer using null hypothesis tests [22]. Concretely, we wish to ascertain, based on the experiments, what is the probability that *MixBoost* is better than SWIM for data augmentation. We repeat the setup described in Sharma et al. [22], where, based on the assumption of the Dirichlet process, the posterior probability for the Bayesian signed test is calculated as a mixture of Dirac deltas centered on the observation.

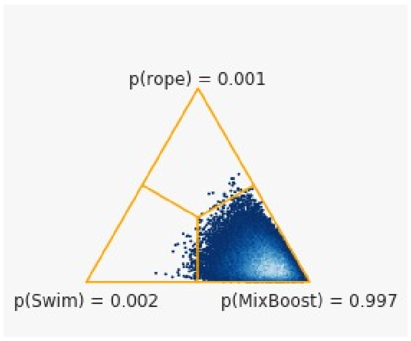
Figure 5 shows the posterior plots of the Bayesian signed test for the three down-sampled datasets. The posteriors are calculated with the prior parameter of the Dirichlet as $s = 0.5$ and $z_0 = 0$ as suggested by Sharma et al. [22]. The posterior plots report the samples from the posterior (blue cloud of points), the simplex (the large triangle), and three regions. The three regions of the triangle denote the following, (1) The region in the bottom left of the triangle indicates the case where it is more probable that SWIM is better than *MixBoost*. (2) The region in the bottom right of the triangle indicates the opposite, i.e., when it is more probable that *MixBoost* is better than SWIM. (3) The region at the top of the triangle indicates that it is likely that neither method is better. The closer the points are to the sides of the triangle, the larger is the statistical difference between methods. Figure 5 shows that the probability that *MixBoost* is better than SWIM for data augmentation is high for all three down-sampled training datasets. For all three plots, the point cloud is concentrated in the bottom right triangle, indicating that *MixBoost* is statistically significantly better than SWIM.

Dataset	Baseline	ALT	SWIM	*MixBoost*	
				R-Selection	EW-Selection
Abalone 9-18	0.481	0.612	0.723	0.743 ± 0.03	0.735 ± 0.06
Diabetes	0.259	0.509	0.509	0.701 ± 0.05	0.560 ± 0.04
Wisconsin	0.874	0.956	0.958	0.960 ± 0.02	0.969 ± 0.08
Wine Q. Red 4	0.224	0.502	0.535	0.714 ± 0.04	0.815 ± 0.08
Wine Q. White 3v7	0.451	0.572	0.730	0.743 ± 0.06	0.750 ± 0.05
Vowel 10	0.724	0.738	0.812	0.845 ± 0.043	0.854 ± 0.07
Pima Indians	0.276	0.479	0.509	0.700 ± 0.05	0.597 ± 0.04
Vehicle 0	0.534	0.758	0.814	0.900 ± 0.05	0.850 ± 0.02
Vehicle 1	0.541	0.739	0.791	0.700 ± 0.03	0.735 ± 0.03
Vehicle 2	0.450	0.549	0.560	0.880 ± 0.02	0.638 ± 0.03
Vehicle 3	0.402	0.505	0.569	0.651 ± 0.06	0.600 ± 0.03
Ring Norm	0.274	0.933	0.899	0.550 ± 0.04	0.580 ± 0.03
Waveform	0.301	0.701	0.688	0.812 ± 0.03	0.844 ± 0.05
PC4	0.572	0.559	0.611	0.720 ± 0.08	0.737 ± 0.04
PieChart	0.455	0.516	0.576	0.611 ± 0.06	0.741 ± 0.07
Pizza Cutter	0.468	0.506	0.552	0.725 ± 0.05	0.678 ± 0.07
Ada Agnostic	0.451	0.445	0.539	0.690 ± 0.02	0.648 ± 0.03
Forest Cover	0.561	0.554	0.550	0.910 ± 0.05	0.917 ± 0.02
Spam Base	0.440	0.550	0.685	0.872 ± 0.03	0.834 ± 0.05
Mfeat Karhunen	0.274	0.933	0.899	0.888 ± 0.07	0.927 ± 0.05

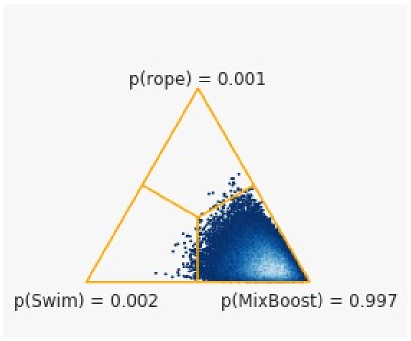
Table 2: Results (mean and standard deviation) with *MixBoost* for data augmentation compared to existing over-sampling methods on the *g-mean* metric. The results correspond to the case where we down-sample the training dataset to have 4 minority class instances (R4). Baseline refers to the case where the binary classifier is trained on the original dataset without data augmentation. ALT is the score of the best performing data augmentation strategy (other than SWIM [22] and *MixBoost*). The best score for each dataset is in bold.



(a) Figure A



(b) Figure B



(c) Figure C

Figure 5: Posteriors for SWIM versus *MixBoost* (left and right vertex) on the datasets with 4, 7 and 10 minority class instances ((a), (b), and (c)) for the Bayesian sign-rank test. A higher concentration of points on one of the sides of the triangle shows that a given method has a higher probability of being statistically significantly better [22]. The top vertex *rope* indicates the case where neither method is statistically significantly better than the other.

4.2 Ablation Studies

We evaluate the importance of the different components of *MixBoost* and its sensitivity to hyper-parameters using ablation studies. To balance the consistency of results and ease of analysis and presentation, we present all results on five datasets. We select these datasets keeping in mind diversity concerning feature dimensionality and the extent of class imbalance. For each study (unless otherwise specified), we augment the training dataset with n synthetic hybrid instances, where n is the number of instances in the training dataset. We sample the interpolation ratio λ from a $Beta(0.5, 0.5)$ distribution. We generate the n synthetic hybrid instances over five *MixBoost* iterations, generating $0.2n$ instances in each *MixBoost* iteration. For *MixBoost*, we use *EW-Selection* for each dataset.

Dataset	**SWIM**	***MixBoost***
Abalone 9-18	0.790 ± 0.08	**0.820** ± 0.06
Diabetes	0.778 ± 0.03	**0.800** ± 0.02
Wisconsin	0.899 ± 0.08	**0.972** ± 0.01
Wine Q. Red 4	0.950 ± 0.03	**0.971** ± 0.03
Wine Q. White 3 vs 7	0.640 ± 0.09	**0.783** ± 0.05
Vowel 10	0.895 ± 0.05	**0.958** ± 0.03
Pima Indians	0.778 ± 0.03	**0.800** ± 0.02
Vehicle 0	0.628 ± 0.04	**0.896** ± 0.07
Vehicle 1	0.592 ± 0.06	**0.750** ± 0.02
Vehicle 2	0.824 ± 0.03	**0.921** ± 0.03
Vehicle 3	0.585 ± 0.05	**0.664** ± 0.03
Ring Norm	0.704 ± 0.24	**0.892** ± 0.08
Waveform	0.828 ± 0.02	**0.869** ± 0.02
PC4	0.686 ± 0.09	**0.794** ± 0.03
PieChart	0.661 ± 0.06	**0.743** ± 0.08
Pizza Cutter	0.662 ± 0.08	**0.735** ± 0.04
Ada Agnostic	0.671 ± 0.05	**0.798** ± 0.02
Forest Cover	0.881 ± 0.07	**0.970** ± 0.02
Spam Base	0.769 ± 0.12	**0.835** ± 0.07
Mfeat Karhunen	0.980 ± 0.01	**0.992** ± 0.00

Table 3: ROC-AUC scores (mean and standard deviation) for *MixBoost* and SWIM for data augmentation. The best score for each dataset is highlighted in bold.

Dataset	***MixBoost-1-Iter***	***MixBoost***
Pima Indians	0.491 ± 0.02	**0.597 ± 0.04**
Waveform	0.843 ± 0.04	**0.844 ± 0.05**
PC4	0.613 ± 0.08	**0.737 ± 0.04**
Piechart	0.721 ± 0.02	**0.741 ± 0.07**
Forest Cover	0.910 ± 0.00	**0.917 ± 0.02**

Table 4: *g-mean* scores (mean and standard deviation over 30 runs) for the single step and iterative variants of *MixBoost*. For all selected datasets, the iterative variant of *MixBoost* outperforms the single step one.

4.2.1 *Impact of sampling instances iteratively in MixBoost.*

In this experiment, we evaluate the importance of generating synthetic hybrid instances iteratively. We create a variant of *MixBoost*, called *MixBoost-1-Iteration* (denoted by *MixBoost-1-Iter*), where the n synthetic instances are generated in a single iteration. This single-step generation is in contrast to *MixBoost*, where the n instances are generated over five iterations, $0.2n$ at each iteration. We augment the training dataset with the generated instances and train the binary classifier on the augmented dataset. Table 4 shows the results. We see that the iterative variant of *MixBoost* outperforms the single-step variant for each dataset. This result illustrates the importance of the iterative element of *MixBoost*.

4.2.2 *Impact of reducing the Number of Minority Class Instances.*

Since *MixBoost* uses an intelligent selection technique for generating synthetic hybrid instances, we want to study whether it outperforms existing approaches with a fewer number of minority class training instances. We generate variations of the training dataset

Dataset	**SWIM**	***MixBoost***		
	min_{ct}=4	min_{ct}=2	min_{ct}=3	min_{ct}=4
Pima Indians	0.499 ± 0.13	0.534 ± 0.08	0.616 ± 0.02	**0.597 ± 0.04**
Waveform	0.652 ± 0.03	0.693 ± 0.06	0.727 ± 0.03	**0.844 ± 0.05**
PC4	0.661 ± 0.06	0.593 ± 0.06	0.688 ± 0.02	**0.737 ± 0.04**
PieChart	0.612 ± 0.05	0.533 ± 0.13	0.579 ± 0.09	**0.741 ± 0.07**
Forest Cover	0.522 ± 0.03	0.452 ± 0.05	0.643 ± 0.03	**0.917 ± 0.02**

Table 5: *g-mean* scores (mean and standard deviation over 30 runs) for data augmentation using *MixBoost* and SWIM as we reduce the number of minority class instances (min_{ct}) by down-sampling. *MixBoost* achieves best performance on all the datasets. Additionally, *MixBoost* outperforms SWIM on 4/5 datasets using half (2 vs 4) the minority class instances.

with 2 and 3 minority class instances. We then run *MixBoost* on these variations and train a classifier on the augmented dataset. Table 5 shows the results. *MixBoost* achieves better results than SWIM while using a fewer number of minority class training examples. For instance, for the PC4 dataset, *MixBoost* outperforms SWIM using half the number of minority class instances (2 vs. 4).

4.2.3 *Selecting different distributions for sampling λ.*

For sampling the interpolation ratio, λ, we experiment with Linear and Non-Linear Probability Density Functions (PDFs). Table 6 shows the results. On all datasets, using a non-linear PDF ($Beta(\alpha, \alpha)$ with $\alpha = 0.5$) in the *Mix* step leads to better classifier performance.

Dataset	**Uniform**	**Beta**
Pima Indians	0.389 ± 0.08	**0.597 ± 0.04**
Waveform	0.661 ± 0.01	**0.844 ± 0.05**
PC4	0.242 ± 0.01	**0.737 ± 0.04**
Piechart	0.312 ± 0.07	**0.741 ± 0.07**
Forest Cover	0.629 ± 0.01	**0.917 ± 0.02**

Table 6: *g-mean* scores (mean and standard deviation over 30 runs) when we sample λ from different distributions. For all datasets, sampling λ from the *Beta(0.5,0.5)* distribution leads to the best performance.

4.2.4 *Impact of Generating Hybrid Instances.*

Existing approaches to augment training data over-sample by generating synthetic minority class instances. *MixBoost* generates synthetic hybrid instances that have elements of both the minority and majority class. Concretely, in the *Mix* step, we generate instances where the target class vector is in-between the majority class vector [1, 0] and the minority class vector [0, 1]. Alternatively, in the *Mix* step, we could have generated synthetic instances where the target class vector corresponded to either the majority or the minority class. To evaluate these alternatives, we create a variant of *MixBoost* in which we assign either the majority class vector (when $\lambda > 0.5$) or the minority class vector (when $\lambda <= 0.5$) to the generated instance. We label this variant *MixBoost-OneHot*. Table 7 shows the results comparing *MixBoost* to *MixBoost-OneHot*. We observe that generating hybrid instances with in-between target class labels (from *MixBoost*) outperforms generating instances with either a majority or a minority class label (from *MixBoost-OneHot*). We posit that training with hybrid instances that are not explicitly attributed to either class enables the learning of more robust decision boundaries by helping regulate the confidence of the classifier away from the training distribution.

Dataset	*MixBoost-OneHot*	*MixBoost*
Pima Indians	0.458 ± 0.03	**0.597 ± 0.04**
Waveform	0.831 ± 0.01	**0.844 ± 0.05**
PC4	0.698 ± 0.07	**0.737 ± 0.04**
Piechart	0.566 ± 0.06	**0.741 ± 0.07**
Forest Cover	0.898 ± 0.02	**0.917 ± 0.02**

Table 7: *g-mean* scores (mean and standard deviation over 30 runs) comparing *MixBoost* to a variant where we generate non-hybrid instances (*MixBoost-OneHot*) in the *Mix* step. For all considered datasets, generating hybrid instances improves the performance of the binary classifier.

4.2.5 Classifier performance as a function of the number of generated synthetic hybrid instances. *MixBoost* generates synthetic instances iteratively. For each iteration, the classifier is retrained using all instances generated up to this iteration. We expect that the marginal gain of generating instances should decrease as more instances are generated. Figure 6 validates our expectation. We see that the performance of the classifier plateaus as *MixBoost* adds more synthetic instances to the training dataset. This plateau in performance is important for the practical deployment of *MixBoost*.

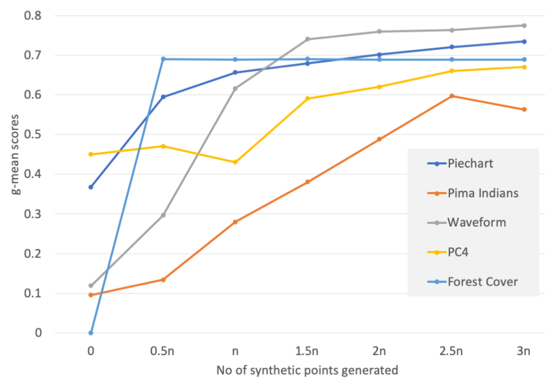


Figure 6: Variation in *g-mean* scores (averaged over 30 runs) for *MixBoost* as we increase the number of generated synthetic hybrid instances. n represents the total number of training instances in the original dataset. The different color lines indicate different datasets. The gain of generating an additional $0.5n$ synthetic instances is initially high and then falls gradually as the number of generated instances increases.

4.2.6 Classifier performance as a function of the number of minority class instances. *MixBoost* mixes an instance of the minority class with an instance of the majority class to create a synthetic hybrid instance. We expect the quality of the generated synthetic instances to improve as we increase the number of minority class instances in the training dataset. The quality of generated instances is measured by the classifier performance. Figure 7 illustrates that classifier performance increases as we increase the number of minority class instances in the training data.

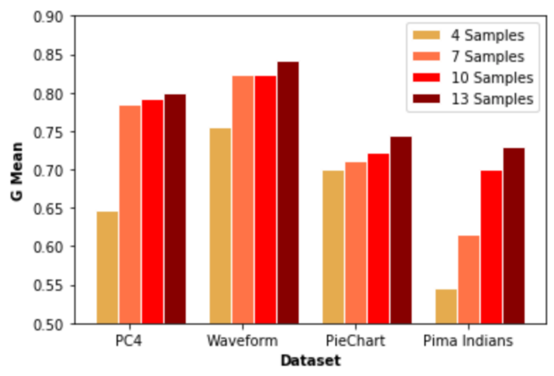


Figure 7: Variation in *g-mean* scores (averaged over 30 runs) for different number of minority class instances. *MixBoost* was used to augment the training dataset in each case. For each dataset, the classifier performance increases with an increase in the number of minority class instances. This increase is more pronounced when we go from 4 to 7 minority samples, after which the classifier performance plateaus.

5 RELATED WORK

There are broadly two categories of approaches for dealing with classification problems on imbalanced datasets. First, sampling-based approaches and second, cost-based approaches [10, 28]. In this paper, we focus on sampling-based approaches. The most straightforward re-sampling strategies are Random under Sampling (RUS), and Random over Sampling (ROS). They balance the class distribution by either randomly deleting instances of the majority class (RUS) or randomly duplicating instances of the minority class (ROS). However, deleting instances from the training dataset can lead to a loss of information. Further, duplicating instances of the minority class does not add any new information.

SMOTE (Synthetic Minority Oversampling Technique) [6] alleviates these shortcomings. SMOTE balances the class distribution by generating synthetic data instances. SMOTE generates a synthetic instance by interpolating the k-nearest neighbors of a minority class instance in the training data. SMOTE has proved useful in a variety of contexts [6, 7, 14]. However, since SMOTE generates synthetic instances using only minority class instances, the generated instances lie within the convex hull of the minority class distribution. Further, because it does not use majority class instances, SMOTE can increase the overlap between classes. Therefore, if the classes are extremely imbalanced, then SMOTE can degrade classifier performance. Extensions of SMOTE [11] add a post-processing step that tries to remove generated instances that might degrade the performance of the classifier. These methods incorporate the majority class distribution into the post-processing step of the data augmentation process. For instance, Adaptive Synthetic Oversampling

(ADASYN) [17], borderline SMOTE [14], and Majority Weighted Minority Oversampling [16] use majority class instances present in the neighborhood of generated instances for post-processing. However, these methods use only the minority class distribution when generating synthetic instances. Therefore, if the number of minority class instances is very small, then the quality of generated instances and subsequently, the performance of the classifier is degraded. Abdi and Hashemi [1] create a variant of SMOTE by using the Mahalanobis distance (instead of Euclidean distance) for generating synthetic instances. However, as with SMOTE, they generate instances using only minority class data. SWIM [22] uses information from the majority class to generate synthetic instances. The authors show how SWIM outperforms existing state-of-the-art methods across several benchmark datasets at extreme levels of class imbalance.

Tanaka and Aranha [25] approach the problem of generating synthetic instances from a different direction. They use a Generative Adversarial Network (GAN) trained on minority class data to generate synthetic training instances. However, their method is unable to consistently outperform existing state-of-the-art methods(Tanaka and Aranha [25], see results). Further, training a GAN for each dataset requires significant compute and hyper-parameter tuning. For these reasons, we have not compared *MixBoost* to their method.

6 CONCLUSION

In the present work, we tackle the problem of binary classification on extremely imbalanced datasets. To this end, we propose *MixBoost*, a technique for synthetic iterative oversampling. *MixBoost* intelligently selects and then combines instances from the majority and minority classes to generate synthetic hybrid instance. We show that *MixBoost* outperforms existing methods on several benchmark datasets. We evaluate the impact of the different components of *MixBoost* using ablation studies.

REFERENCES

[1] Lida Abdi and Sattar Hashemi. 2015. To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE transactions on Knowledge and Data Engineering* 28, 1 (2015), 238–251.

[2] Ali Al-Shahib, Rainer Breitling, and David Gilbert. 2005. Feature selection and the class imbalance problem in predicting protein function from sequence. *Applied Bioinformatics* 4, 3 (2005), 195–203.

[3] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. 2014. A Bayesian Wilcoxon signed-rank test based on the Dirichlet process.. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 32. JMLR.org, 1026–1034. http://dblp.uni-trier.de/db/conf/icml/icml2014.html#BenavoliCMZR14

[4] Kwabena Ebo Bennin, Jacky Keung, Passakorn Phannachitta, Akito Monden, and Solomon Mensah. 2017. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering* 44, 6 (2017), 534–550.

[5] Jonathan Burez and Dirk Van den Poel. 2009. Handling class imbalance in customer churn prediction. *Expert Systems with Applications* 36, 3 (2009), 4626–4636.

[6] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[7] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. 2003. SMOTEBoost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*. Springer, 107–119.

[8] Anonymous MixBoost Code. 2019. MixBoost-Code. https://github.com/user1332/MIXBOOST-code.

[9] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[10] Charles Elkan. 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 973–978.

[11] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. 2018. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research* 61 (2018), 863–905.

[12] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 4 (2011), 463–484.

[13] Salvador García and Francisco Herrera. 2009. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation* 17, 3 (2009), 275–306.

[14] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 878–887.

[15] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 878–887.

[16] H He, Y Bai, E Garcia, and S ADASYN Li. 2008. Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks, 2008 (IEEE World Congress on Computational Intelligence).

[17] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 1322–1328.

[18] Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, Vol. 97. Nashville, USA, 179–186.

[19] Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, Vol. 97. Nashville, USA, 179–186.

[20] S Miri Rostami and M Ahmadzadeh. 2018. Extracting predictor variables to construct breast cancer survivability model with class imbalance problem. *Journal of AI and Data Mining* 6, 2 (2018), 263–276.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[22] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar Zaiane, and Nathalie Japkowicz. 2018. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 447–456.

[23] Cecilia Summers and Michael J. Dinneen. 2018. Improved Mixed-Example Data Augmentation. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), 1262–1270.

[24] Cecilia Summers and Michael J Dinneen. 2019. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1262–1270.

[25] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. 2019. Data Augmentation Using GANs. *arXiv preprint arXiv:1904.09135* (2019).

[26] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Between-Class Learning for Image Classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), 5486–5494.

[27] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5486–5494.

[28] Huihui Wang, Yang Gao, Yinghuan Shi, and Hao Wang. 2016. A fast distributed classification algorithm for large-scale imbalanced data. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1251–1256.

[29] Shuo Wang, Leandro L Minku, and Xin Yao. 2014. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2014), 1356–1368.

[30] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).