Learning from Massive Highly Imbalanced Data via Hybrid-sampling with Self-paced Curriculum

***Abstract*—Data imbalance indeed poses a significant challenge in machine learning. Various resampling based methods are proposed to handle this problem. Among these methods, hybrid-sampling is well-known for addressing the imbalance issue by utilizing over- and under-sampling mechanisms. However, when dealing with massive highly imbalanced data, traditional hybrid-sampling methods suffer from ineffective noise reduction and poor performance. We analyze the underlying design concept of these methods and conduct detailed experiments to study their characteristics, which indicates that in the over-sampling phase of hybrid-sampling methods, generating substantial samples with massive highly imbalanced data hinders the effectiveness of under-sampling in noise reduction and leads to limited, even degraded performance of classifier. To tackle these issues, we propose a novel hybrid-sampling based ensemble learning framework, `SCHE`, that generates and selects reasonable amount of informative samples to build a strong ensemble with a self-paced curriculum learning fashion. Taking dynamic learning state and local data distribution characteristics into consideration, our framework can adapt to massive highly skewed data with complex pattern and cooperate with any canonical classifiers. Experiments on various real-world imbalanced datasets demonstrate that `SCHE` achieves superior performance with reasonable computation cost. Our code is available at** `https://github.com/zxjbibobibobi/SCHE`.

***Index Terms*—imbalance learning, hybrid-sampling, ensemble learning, imbalanced classification**

I. Introduction

Class imbalance is widely observed in research domains like bot detection [1], intelligent fault diagnosis [2] and medical diagnosis [3], referring to an uneven representation of minority and majority class. A larger quantity of majority samples can mislead the classifier into disproportionately focusing on the majority class, diminishing its performance on minority class, which usually carries the concepts with greater interests [4]. Even worse, when dealing with massive highly imbalanced data, classifiers' performance may further degrade.

Existing methods address the imbalance issue by leveraging reweighting [5]–[7] as well as resampling [8]–[10] mechanisms and achieve noticeable success. However, these methods struggle with massive highly imbalanced data. Reweighting based methods have been criticised for requiring assistance from domain experts knowledge. In light of recent research, over-sampling is a double-edged sword for handling imbalanced data [3], [11]. Despite alleviating the negative impact of data imbalance by generating minority samples, they suffer from noise introduction and significant computation cost. Also, under-sampling methods fail to achieve satisfactory performance due to losing information from majority class [12].

Considering the strengths and limitations of over- and under-sampling, hybrid-sampling methods address the imbalance issue from dual sampling perspectives [13], [14]. Hybrid-sampling methods balance the dataset by over-sampling and apply under-sampling to remove noise samples introduced in over-sampling phase. The core idea of traditional hybrid-sampling methods can be summarized as "*balancing through over-sampling and denoising through under-sampling*". This underlying design concept mitigates over-sampling methods' limitations and yield decent performance.

Nevertheless, traditional hybrid-sampling methods become impractical with massive highly imbalanced data. Because of the extreme skew distribution of minority and majority class, hybrid-sampling methods generate substantial minority samples in over-sampling phase, which inevitably introduces a fairly large amount of noise samples. In under-sampling phase, eliminating these noise samples is inefficient, costly, and challenging due to their large amount. Even if it were possible to denoise effectively, why bother introducing so much noise in the beginning? To conclude, the inherent flaw of hybrid-sampling in massive high imbalanced application scenario arises because of the substantial amount of generated samples. We conclude that existing hybrid-sampling methods potentially encounter two significant challenges when dealing with massive highly imbalanced data:

- Significant computation cost in over-sampling phase for generating substantial minority samples.
- Disturbed data distribution and difficult noise reduction caused by substantial generated minority samples.

Henceforth, we adopt a different design concept than traditional hybrid-sampling methods and propose a novel hybrid-sampling based ensemble framework for massive highly imbalanced data classification. We argue that only an *appropriate* number of *informative* minority samples should be generated in over-sampling phase. And the imbalance issue should not be addressed merely through over-sampling, but also under-sampling. Following this, *informative* majority samples are selected to construct a balanced training set with limited size. Due to the limited number of majority samples in the balanced training set, which prevents the model from comprehensively learning the majority class, we draw inspiration from curriculum learning [15], [16] and naturally integrate this hybrid-sampling process into an ensemble learning framework with a self-paced curriculum fashion. The ensemble can finally obtain satisfactory performance from multiple training set with limited size.

In this paper, we propose a *Self-paced Curriculum Hybrid-sampling based Ensemble* (abbreviated as SCHE) following a different design concept than traditional hybrid-sampling. SCHE aims at training an ensemble model by applying hybrid-sampling in line with self-paced curriculum mechanism. Initially, SCHE determines the number of generated minority samples based on clustering results of minority class and calculates the size of training sets using size scheduler. In each iteration, SCHE calculates instance hardness by weighting the dynamic learning state of the ensemble model (i.e., dynamic hardness) and the local data distribution characteristics (i.e., local hardness) via weight scheduler. With the guidance of instance hardness, an easy-to-hard curriculum for training base classifiers is designed. In the end, a base classifier is trained and added to the ensemble. We also revisit the key components of the classical over-sampling method, SMOTE [8], and integrate them into SCHE's cluster-wise over-sampling process. In summary, this paper contributes in three aspects:

- We analyze the inherent flaw of traditional hybrid-sampling methods and illustrate their limitations when dealing with massive highly imbalanced data through detailed experiments.
- Based on the analysis conducted on hybrid-sampling methods, we propose *Self-paced Curriculum Hybrid-sampling based Ensemble* (SCHE) for massive highly imbalanced data classification. SCHE can stably boost base classifier's performance while obtaining an reasonable trade-off between model performance and training cost.
- We introduce the concept of instance hardness, a weighted combination of dynamic learning state of ensemble model (i.e., dynamic hardness) and local data distribution characteristics (i.e., local hardness). With the guidance of instance hardness, SCHE can automatically adapt the hybrid-sampling strategy and optimize base classifier's performance in a data/model-specific way.

II. Related Work

Imbalanced data classification remains to be a significant challenge in machine learning [4]. In our work, we focus on the fundamental binary classification problem, which is more preferred by researchers in imbalance learning [3].

Existing hybrid-sampling methods address imbalance issue by over-sampling, mostly using SMOTE [8], and utilize under-sampling methods based on different heuristics to reduce the introduced noise. TLinks [17] and ENN [18] are respectively used to clean noise samples in SMOTETomek [14] and SMOTEENN [13]. SMOTE−RkNN [19] utilizes reverse kNN to detect noise samples with significantly higher probability density in the other class. When dealing with massive highly imbalanced data, these hybrid-sampling methods suffer from large computation cost and ineffective noise reduction due to their strategy of *"balancing through over-sampling and denoising through under-sampling"*as we mentioned and analyzed in Introduction I.

Ensemble methods integrate resampling mechanism with ensemble training procedure to handle imbalanced data. Some representative over-sampling based ensemble methods [20], [21] utilize SMOTE [8] to generate minority samples and combine base classifiers through bagging [22] or boosting [23]. When dealing with massive highly imbalanced data, these over-sampling based ensemble methods suffer from overfitting and large computation cost. Though under-sampling based ensemble methods [24] like RUSBoost [25] and Cascade [26] reduce training cost significantly compared to over-sampling based methods, they potentially cause over/underfitting.

Aimed at learning robust and adaptive ensemble model from massive highly imbalanced data, some research [27], [28] introduces learning-based metrics to guide the sampling process. Among recent work, the most prominent one is SPE [28], an under-sampling based ensemble method. It introduces classification hardness together with hardness harmonization to select majority samples with different learning difficulty. Essentially, SPE adopts a classical curriculum learning fashion, *self-paced learning* [16], to provide an easy-to-hard learning process for ensemble model. Though SPE demonstrates outstanding performance with simple heuristic, it overlooks the integration of over-sampling mechanism. Hence, there's potential to further boost under-sampling based ensemble imbalanced learning framework by introducing over-sampling.

To summarize, when dealing with massive highly imbalanced data, traditional hybrid-sampling methods inherently suffer from costly sample generation and ineffective noise reduction due to the massive generated samples. In comparison, SCHE generates highly informative minority samples with much fewer amount to boost classifier's performance. We redesign the core components of SMOTE [8] to minimize the introduction of noise and simultaneously produce highly informative minority samples. Moreover, the training order of samples in SCHE is determined by utilizing learning-based metrics (i.e, dynamic hardness) and predefined criteria (i.e., local hardness). Thus, SCHE adopts a popular curriculum learning fashion, *self-paced curriculum learning* [16] to provide an easy-to-hard curriculum for ensemble training.

III. Proposed Method

A. Overview of SCHE

Fig. 1 demonstrates SCHE's overview. In each iteration, instance hardness is weighted with updated dynamic hardness and constant local hardness via weight scheduler. The size of training set is calculated by size scheduler. Following this, a balanced training set is obtained jointly through self-paced curriculum over-/under-sampling. In the over-sampling phase, we conduct a bin-division operation on each clusters defined by DBSCAN. Then, hardness harmonization mechanism and redesigned SMOTE are applied to generate minority samples cluster-wise. In under-sampling phase, majority samples are selected through hardness harmonization. Finally, a classifier is trained and added to the ensemble.

B. Instance Hardness

We leverage instance hardness, a weighted combination of dynamic and local hardness, to guide the hybrid-sampling

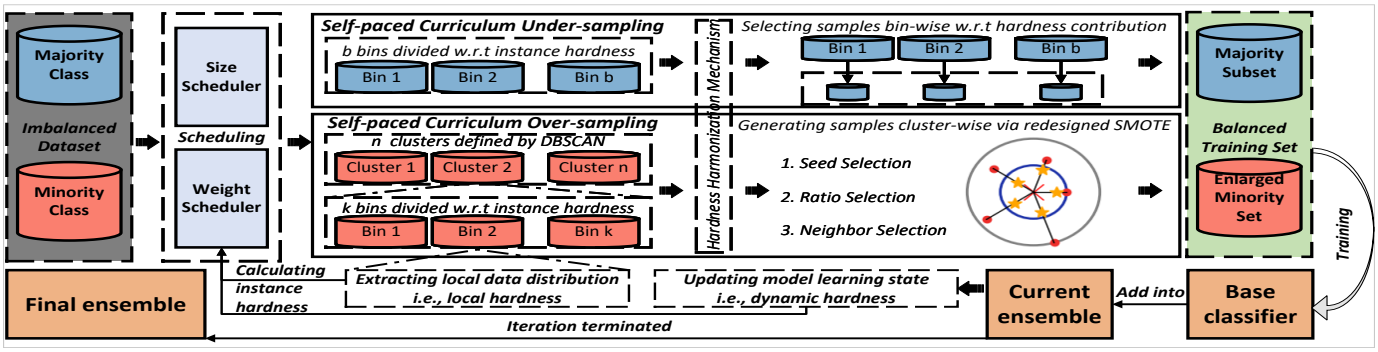


Fig. 1: Overview of Self-paced Curriculum Hybrid-sampling based Ensemble.

process in SCHE. In this section, we first introduce dynamic and local hardness, then further describe their relationships and the advantages of instance hardness in guiding ensemble training with imbalanced data.

Dynamic hardness reflects the error of model's prediction on training samples. In the i^{th} iteration, given the current ensemble classifiers F_i, a dynamic hardness function H, a sample x and its label y, dynamic hardness is denoted as:

$$d\_hardness = H(F_i(x),\ y). \tag{1}$$

In Eq. 1, H can be any decomposable error function like Absolute Error, Squared Error or Cross Entropy [28]. $F_i(x)$ is the output classification probability of input sample x in the i^{th} iteration. As the model's learning state of sample x gets better/worse, corresponding $d\_hardness$ decreases/increases.

Local hardness is another sample-level concept that reflects the class heterogeneity in a sample's local area. The local hardness of each sample is calculated as:

$$l\_hardness = \frac{k^{'}}{k}. \tag{2}$$

In Eq. 2, given a sample, k is the number of its neighbor, and $k^{'}$ is the number of neighbor with different labels than itself.

From Eq. 1 and Eq. 2 we can observe that dynamic hardness is *varying*, *continuous*, and *model-specific* while local hardness is *constant*, *discrete* and *data-specific*. Although these two hardness have distinct features, they both reflect the learning difficulty of samples, based on which we can divide training samples into **safe**, **borderline** and **noise** from a traditional imbalanced learning perspective [3], [4]:

- **Safe** sample has low hardness level with a large population, which makes its total hardness contribution non-negligible. An appropriate number of safe sample benefits the model in acquiring basic classification ability while excessive safe samples hinder the learning pace of model.
- **Borderline** sample has medium hardness level. On the one hand, well-classifying these samples is hard for the current ensemble model. On the other hand, achieving favorable learning state of these samples is feasible.
- **Noise** sample has high hardness level with a relatively small population. Forcing the model to learn from these samples will lead to severe overfitting.

Regardless of the hardness used to classify training samples, we can apply same strategies to process the three resulting categories when training ensemble model with imbalanced data. For **safe** samples, a relatively large proportion of them should be kept in early iterations to enable the model quickly acquire fundamental learning ability. As training process proceeds, the amount of safe training samples should be gradually reduced to shift the model's attention to harder-to-learn samples. As model's learning state improves, the amount of **borderline** samples should be gradually increased for further performance enhancement. In contrast, the quantity of **noise** samples in each iteration should be minimized as much as possible.

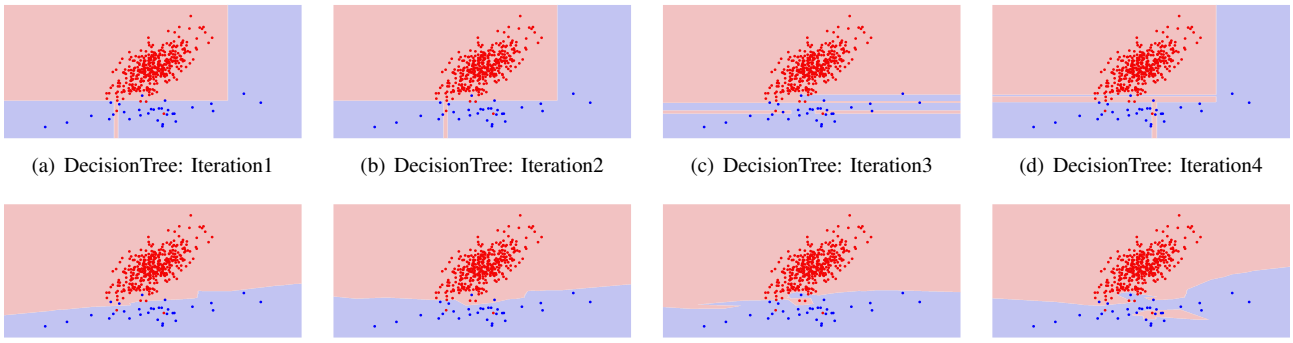
Due to these shared properties and processing strategies, it is reasonable to employ instance hardness, a combination of local hardness and dynamic hardness, to comprehensively reflect the learning difficulty of samples from both model and data perspectives. As the ensemble training process with imbalanced data shown in Fig. 2, ensemble model's decision boundary varies in a data/model-specific way, resulting in variations of sample's learning difficulty highly correlated to dynamic model learning state and local data distribution characteristics. Utilizing instance hardness to guide the ensemble training process with imbalanced data has these advantages:

- Instance hardness describes the difficulty of imbalanced classification task comprehensively, which can't be fully reflect by imbalance ratio.
- Instance hardness establishes the connections between hybrid-sampling strategy, model performance and data distribution. The performance of ensemble can be optimized in a data/model-specific manner.

C. Size Scheduler & Weight Scheduler

Before leveraging instance hardness to guide the hybrid-sampling process in SCHE, two crucial questions should be answered: (I)What's the size of training set in each training iteration? (II)How to weight dynamic and local hardness? We introduce size and weight scheduler to calculate the amount of training samples and instance hardness in each iteration.

1) Size Scheduler: Size scheduler calculates the number of generated minority samples in each iteration to determine the size of balanced training set. Inspired by research related to



(a) DecisionTree: Iteration1 (b) DecisionTree: Iteration2 (c) DecisionTree: Iteration3 (d) DecisionTree: Iteration4

(e) KNN: Iteration1 (f) KNN: Iteration2 (g) KNN: Iteration3 (h) KNN: Iteration4

Fig. 2: As ensemble model's decision boundary changes over iterations, the learning difficulty of samples varies in a data/model-specific manner. Respectively, (a)(b)(c)(d) and (e)(f)(g)(h) demonstrate this process when DT [29] and KNN [30] are used as base classifiers for Bagging [22].

within-class imbalance [31]–[33], we first determine the *appropriate* number A of generated minority samples by utilizing clustering methods. We apply DBSCAN [34] to cluster the minority class because it's robust to complex data distribution and can automatically determine the number of clusters. After obtaining c clusters $c_1, c_2, \ldots, c_c$, A is calculated as:

$$A = \sum_{i=1}^{c} (max(|c_1|, |c_2|, \ldots, |c_c|) - c_i). \tag{3}$$

These clusters represent different sub-concepts in minority class. Classifiers may be biased towards the largest sub-concept. Balancing sub-concepts improves the data distribution from a within-class perspective.

Given A and the number of iterations n, size scheduler uses a scheduler function SF to control the generating speed of minority samples. The scheduler function can be defined as a convex, linear, concave or composite function [35], which returns values monotonically increasing from 0 to 1. In SCHE, we adopt a scheduler function with constant learning speed:

$$SF^{size}_{linear}(i) = \frac{i}{n}, 0 \leq i \leq n. \tag{4}$$

With SF^{size}_{linear}, the number of minority samples to be generated in the i^{th} iteration is calculated as:

$$A_i = A * (SF^{size}_{linear}(i) - SF^{size}_{linear}(i-1)). \tag{5}$$

Correspondingly, denoting the majority and minority class in the original training set D as N and P, the size of training set in the i^{th} iteration is calculated as:

$$|D_i| = |N_i| + |P_i| = 2 \times (|P| + A_i). \tag{6}$$

2) Weight Scheduler: Weight scheduler determines the weights for dynamic and local hardness in each iteration. Instance hardness in the i^{th} iteration is weighted as:

$$\begin{aligned} i\_hardness_i = & l\_hardness_i \times l\_weight_i \\ & + d\_hardness_i \times d\_weight_i, \end{aligned} \tag{7}$$

$$l\_weight_i + d\_weight_i = 1. \tag{8}$$

We aim for instance hardness to mainly reflect the model's learning state in early iterations. As model's learning ability improves, we hope that instance hardness could further guide the optimization of model performance with local data distribution characteristics. Thus, we utilize a linear scheduler function to control the variation of dynamic hardness:

$$SF^{d\_weight}_{linear}(i) = 1 - \frac{i}{2n}, 0 \leq i \leq n. \tag{9}$$

$SF^{d\_weight}_{linear}$ returns value decreasing from 1 to 0.5 monotonically. Accordingly, l_weight increase from 0 to 0.5. By weighting dynamic and local hardness, SCHE adopts the *self-paced curriculum* [16] learning fashion which utilizes learning-based metrics (i.e., dynamic hardness) and predefined criteria (i.e., local hardness) jointly to train an ensemble model.

D. Hybrid-sampling Process

With instance hardness and schedulers, SCHE constructs balanced training set through hybrid-sampling. This section introduces the over- and under-sampling phase in SCHE.

1) Over-sampling Phase: We aim to leverage the classical over-sampling technique, SMOTE [8], into our learning framework while avoiding noise introduction and generating informative samples as much as possible. SMOTE selects three key components to generate synthetic samples $x_{synthetic}$: (I)a seed sample x_{seed} as the sampling center, (II)a neighbor sample $x_{neighbor}$ from the k-nearest neighbors of x_{seed}, (III)$ratio \in [0, 1]$ for determining the position of linear interpolation when generating $x_{synthetic}$. $x_{synthetic}$ is generated as:

$$x_{synthetic} = x_{seed} + ratio * (x_{seed} - x_{neighbor}). \tag{10}$$

To generate highly informative samples while minimizing noise introduction, We tailor the selection of x_{seed} and $ratio$ by leveraging the distinctive features of SCHE.

First, considering that the amount of generated minority samples is calculated based on the result of DBSCAN clustering, we utilize this clustering structure to conduct a cluster-wise over-sampling process in SCHE. When selecting x_{seed}

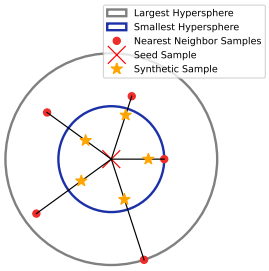


Fig. 3: Illustration of how SCHE selects $x_{neighbor}$ and performs linear interpolation with upper-bounded $ratio$.

in a cluster to generate new samples, only $x_{neighbor}$ within the same cluster is eligible for linear interpolation in Eq. 10. Thus the synthetic sample does not leave the geometric space of that cluster so as to avoid introducing samples that do not conform to the original cluster structure.

Besides, we select x_{seed} with the expectation that generating samples around them will gradually improve the learning state of model. Therefore, we conduct hardness harmonization mechanism [36] with a bin-division operation, which proves to be effective in training imbalanced ensemble model [27], [28], to calculate sampling weights for each bin according to its instance hardness contribution.

Moreover, as we expect $x_{synthetic}$ can well reflect the local distribution characteristics of x_{seed}, we only allow $x_{synthetic}$ to be generated within the smallest hypersphere formed by the x_{seed} and its neighbors. By controlling the upper bound of $ratio$, the generation region is limited.

The details of over-sampling phase in SCHE, i.e., *Self-paced Curriculum Over-sampling* are shown in Algorithm 1. The parameter k is used for calculating local hardness, as well as the number of k-nearest neighbors. B_ℓ (line 4) is defined as:

$$B_\ell = \{(x,y)|l\_hardness(x) = \frac{\ell-1}{k}\}, \forall \ell = 1,\ldots,k$$

We consider samples with $l\_hardness$ equals to 1 as noise, so these samples won't participate in the sampling process. As a result, we divide minority samples into k bins.

We select x_{seed} with hardness harmonization (line 5-6) to generate samples around them to enhance ensemble model's learning ability. Besides, We randomly select $x_{neighbor}$ (line 9) to maintain sampling diversity. As shown in Fig. 3, a simple illustration of 2-D samples, x_{seed} is the center sample with 5 nearest neighbors. Given x_{seed}, $\min(dist(x_{seed}, x_{neighbor\_m}))$ is the radius of the smallest hypersphere (the blue circle) formed by x_{seed} and its neighbors. Synthetic samples will only be generated within this region whichever $x_{neighbor}$ is selected. This is because we set the upper bound of $ratio$ as $\frac{dist_{min}}{dist(x_{seed},x_{neighbor})}$ instead of 1 (line 11) to limit the range of linear interpolation. Note that SMOTE allows a larger region (the grey circle in Fig. 3) for sample generation, which may resulting in $x_{synthetic}$ being distant from x_{seed}, thus failing to accurately reflect x_{seed}'s local data distribution characteristics.

Algorithm 1: Self-paced Curriculum Over-sampling

Input: Minority set $\mathcal{P}$, number of minority samples to be generated in current iteration A_i, number of bins in minority class k, instance hardness in current iteration $i\_hardness$, clustering results of minority set $cluster\_label$

1 $\mathcal{P}_i \Leftarrow \mathcal{P}$
2 **for** c *in* $cluster\_label$ **do**
3 $P_c^{'} \Leftarrow$ samples with cluster label c in $\mathcal{P}$
4 Cut $P_c^{'}$ into k bins w.r.t $l\_hardness$: $B_1, \cdots, B_k$
5 Average instance hardness contribution in ℓ-th bin: $h_\ell = \sum_{s\in B_\ell} i\_hardness_\ell / |B_\ell|,\ \forall \ell = 1,\ldots,k$
6 Normalized sampling weight of ℓ-th bin: $w_\ell = \frac{1/h_\ell}{\sum_m 1/h_m}, \forall \ell = 1,\ldots,k$
7 Bootstrap $A_i \cdot w_\ell$ seed samples x_{seed} from ℓ-th bin
8 **for** *each* x_{seed} *in* ℓ*-th bin* **do**
9 Randomly select $x_{neighbor}$ from k-nearest neighbors in minority class
10 Calculate the minimum distance between x_{seed} and each k-nearest neighbor: $dist_{min} = \min(dist(x_{seed}, x_{neighbor}))$
11 Calculate the interpolation ratio: $ratio = random(0, \frac{dist_{min}}{dist(x_{seed},x_{neighbor})})$
12 Generate new $x_{synthetic}$ with $x_{seed}, x_{neighbor}, ratio$ by (10)
13 $\mathcal{P}_i \Leftarrow \mathcal{P}_i \cup x_{synthetic}$
14 **end**
15 **end**
16 **return** *enlarged minority set* $\mathcal{P}_i$

Algorithm 2: Self-paced Curriculum Under-sampling

Input: Majority set $\mathcal{N}$, size of enlarged minority set $|\mathcal{P}_i|$, number of bins in majority b, instance hardness in current iteration $i\_hardness$

1 Cut $\mathcal{N}$ into b bins w.r.t. $i\_hardness$: $B_1, \cdots, B_b$
2 Update self-paced factor $\alpha = tan(\frac{i\pi}{2n})$
3 Average instance hardness contribution in ℓ-th bin: $h_\ell = \sum_{s\in B_\ell} i\_hardness_\ell / |B_\ell|,\ \forall \ell = 1,\ldots,b$
4 Normalized sampling weight of ℓ-th bin: $w_\ell = \frac{1/(h_\ell+\alpha)}{\sum_m 1/(h_m+\alpha)}, \forall\ m = 1,\ldots,b$
5 Undersample ℓ-th bin and obtain majority subset $\mathcal{N}_l$, $|\mathcal{N}_l| = |\mathcal{P}_i| \cdot w_\ell, \forall \ell = 1,\ldots,b$
6 **return** *majority subset* $\mathcal{N}_i = \mathcal{N}_1 \cup, \ldots, \cup \mathcal{N}_b$

2) Under-sampling Phase: After over-sampling the minority class, *Self-paced Curriculum Under-sampling* (as shown in Algorithm 2) is conducted to obtain a majority subset w.r.t the size of enlarged minority set $\mathcal{P}_i$. b is the number of bins when dividing majority set. The B_ℓ here (line 1) is defined as:

$$B_\ell = \{(x,y)|i\_hardness(x) \in [\frac{\ell-1}{b}, \frac{\ell}{b})\}, \forall \ell = 1,\ldots,b$$

Similar to the definition of B_ℓ in over-sampling phase, we manage to divide samples into several intervals according to the three categories of samples mentioned in section III-B "softly". To be clear, we update the self-paced factor α (line 2) the same as SPE [28] to accelerate training speed and calculate sampling weights through hardness harmonization (line 3-4).

E. Detailed Process of SCHE

Finally, we present the detailed process of *Self-paced Curriculum Hybrid-sampling based Ensemble* in Algorithm 3. We use D to denote the training set. $cluster\_label$ is the clustering results of $\mathcal{P}$ obtained by DBSCAN. $l\_hardness$ is calculated at once when initializing (line 5). In i^{th} iteration, weights of $l\_hardness$ and $d\_hardness$ are updated to calculate $i\_hardness$ (line 8-10). Then *Self-paced Curriculum Over-/Under-sampling* are applied to construct a balanced training set according to the dynamic learning state of $F_i(x)$ and the local data distribution characteristics (line 11-12).

Algorithm 3: SCHE

Input: Training set D, hardness function H, scheduler function SF_s, SF_w, base classifier f, number of bins in majority class b, number of bins in minority class k, number of base classifier n

1 **Initialize:** $\mathcal{P} \Leftarrow$ minority in $\mathcal{D}$, $\mathcal{N} \Leftarrow$ majority in $\mathcal{D}$
2 Obtain clusters results $cluster\_label = (c_1 \dots c_d, noise)$ of $\mathcal{P}$ by DBSCAN
3 Calculate the total amount of generated samples A by balancing each cluster by (3).
4 Train f_0 using random under-sampled majority subsets $\mathcal{N}_0'$ and $\mathcal{P}$, where $|\mathcal{N}_0'| = |\mathcal{P}|$.
5 Calculate $l\_hardness$ by (2)
6 **for** *i=1 to n* **do**
7 Ensemble $F_i(x) = \frac{1}{i}\sum_{j=0}^{i-1} f_i(x)$
8 Calculate dynamic hardness $d\_hardness_i$ by (1)
9 Calculate number of generated samples A_i by (5)
10 Update hardness weights and $i\_hardness_i$ by (7)
11 Obtain enlarged minority set: $\mathcal{P}_i \Leftarrow$ Self-paced Curriculum Over-sampling $(\mathcal{P}, A_i, k, i\_hardness_i, cluster\_label)$
12 Obtain under-sampled majority subset: $\mathcal{N}_i' \Leftarrow$ Self-paced Curriculum Under-sampling $(\mathcal{N}, |\mathcal{P}_i|, b, i\_hardness_i)$
13 Training f_i using $\mathcal{P}_i \cup \mathcal{N}_i$
14 **end**
15 **return** *final ensemble* $F(x) = \frac{1}{n}\sum_{m=1}^{n} f_m(x)$

Actually, through preliminary study we found that calculating local hardness for majority class brings unbearable computation cost. Moreover, the distribution of majority samples' local hardness is too skewed to effectively reflect their learning difficulty. Take Credit Fraud for example, out of total 227452 training samples, only 132 samples have a local hardness value greater than 0. Thus we only calculate dynamic hardness, instead of instance hardness for majority class.

IV. Experiment

A. Evaluation Metrics

To appropriately evaluate model performance for binary classification task, we adopt F1-score and G-mean, the harmonic and geometric mean of precision and recall calculated from confusion matrix (TABLE I). Moreover, AUPRC (i.e., area under the precision-recall curve) and MCC (i.e., Matthews correlation coefficient) are also considered.

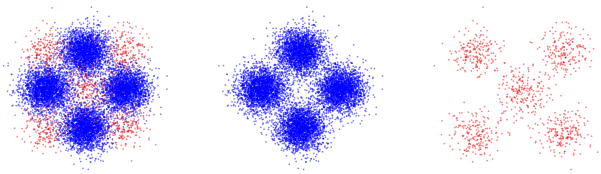
TABLE I: Confusion matrix for binary classification.

Label \ Predict	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

- $Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$
- *F1-score* = $2 \times \frac{Precision \times Recall}{Precision+Recall}$
- *G-mean* = $\sqrt{Precision \times Recall}$
- *MCC* = $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
- *AUPRC = Area Under Precision-Recall Curve*

B. Intuitive Performance Evaluation with Synthetic Datasets

We first conduct experiments on synthetic CheckerBoard datasets as in previous studies [27], [28] to intuitively demonstrate SCHE's performance under varying data distribution.



(a) CheckerBoard (b) Majority Class (c) Minority Class

Fig. 4: Class distribution of CheckerBoard dataset.

1) Experiment Setup: We first create a 3 × 3 CheckerBoard dataset (as shown in Fig. 4) which contains 9 Gaussian components sharing the same covariance matrix to demonstrate SCHE's performance. The overlapping level is controlled by covariance factor. As the value of covariance factor gets higher, the CheckerBoard dataset becomes more overlapped. The imbalance ratio of this dataset is 10, the covariance factor used is 0.8 and the number of minority samples is 1000. Second, we further validate SCHE's effectiveness on CheckerBoard datasets with varying imbalance ratio and overlapping level. We keep the number of minority samples constant and increase the imbalance ratio by enlarging the size of majority class. Each CheckerBoard dataset is split into training set and testing set with a proportion of 80%:20%. All subsequently used datasets are split in this proportion. Our proposed SCHE is first compared with 2 classical hybrid-sampling methods:

- SMOTEENN [13] uses SMOTE [8] to generate minority samples and utilizes ENN [18] for noise reduction.
- SMOTETomek [14] also uses SMOTE to address the imbalance issue, but utilizes TLinks [17] for noise reduction.

We use 7 canonical classifiers (as listed in TABLE II) to cooperate with SCHE and baseline methods. We also list the specific hyper-parameter of each classifier. Note that the number of base classifier in SCHE is 50[1].

TABLE II: STATISTICS OF 7 CANONICAL CLASSIFIERS

Model	Abbreviation	Hyper-parameter
Logistic Regression	LR	None
Decision Tree [29]	DT	max_depth=10
K-Nearest Neighbors [30]	KNN	k_neighbors=5
Support Vector Machine [37]	SVC	C=1000
Adaptive Boosting [38]	AdaBoost	n_estimator=10
Random Forest [39]	RF	n_estimator=10
Gradient Boosting Decision Tree [40]	GBDT	n_estimator=10

2) Results on Synthetic CheckerBoard Datasets: Fig. 5 shows the experimental results on CheckerBoard. We conduct 10 independent runs and demonstrate the mean value of AUPRC. All subsequent listed metrics' value are also the mean of results from 10 independent runs.

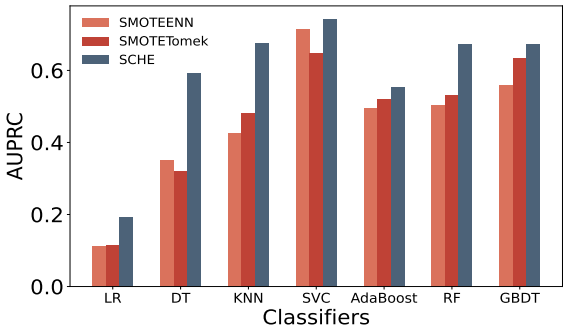
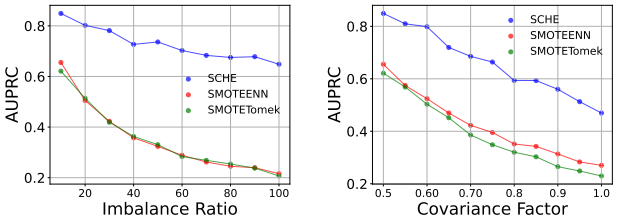


Fig. 5: AUPRC results on CheckerBoard dataset.

Experimental results show that SCHE achieves best performance on 7 classifiers compared with SMOTEENN and SMTOETomek. Further more, we show how AUPRC changes within increasing imbalance ratio and overlapping level of CheckerBoard in Fig. 6 when DT is used as classifier.

* SCHE demonstrates better robustness to increasing imbalance ratio and overlapping level. As data distribution condition gets worse, the performance of SMOTEENN and SMOTETomek quickly degrade while SCHE maintains relatively high performance by a large margin.
* SCHE outperforms two hybrid-sampling baselines because it generates informative minority samples with a limited amount and selects informative majority samples to build a robust ensemble. The massive generated minority samples of SMOTEENN and SMOTETomek ruins the effectiveness of the under-sampling mechanism they utilize, leading to classifier's limited performance.

[1]In our experiments, the value pf parameter k is set as 4 and the number of bins in majority class is set as 10. The dynamic hardness function H is an absolute error function.



(a) Imbalance Level (b) Overlapping Level

Fig. 6: AUPRC variation w.r.t different imbalance ratio and overlapping level. In subplot(a), each dataset's covariance factor is set as 0.8. In subplot(b), each dataset's imbalance ratio is set as 10.

C. Effectiveness on Real-world Small-scale Imbalanced Data

To validate SCHE's effectiveness of tackling real-world imbalanced classification tasks, we introduce 6 small-scale imbalanced datasets loaded from the imbalance-learn package [41]. Although these datasets have relatively low imbalance ratio and small size, they can still be tough tasks for imbalance learning methods because of their complex data distributions.

1) Experiment Setup: We introduce another 3 imbalance learning methods, which contain representative resampling and ensemble methods, for a more comprehensive comparison.

- SMOTE (*Synthetic Minority Over-sampling TechniquE*) [8] generates samples by linear interpolation.
- TLinks [17] identifies and removes 'Tomek Links' to under-sample the majority class.
- SPE (*Self-paced Ensemble*) [28] dynamically under-samples the majority class with the guidance of model's learning state to train a strong ensemble model.

2) Results on Real-world Small-scale Imbalanced Datasets: TABLE III lists the most representative results of each toy dataset. To reduce randomness, the mean and standard variation of 4 metrics in these runs are listed. We use subscript to indicate the number of base classifiers in ensemble model (RF_{10} indicates a Random Forest classifier with 10 DTs as its base classifier). Table III shows us:

* SCHE outperforms all baseline methods using 7 classifiers on 6 toy datasets with various imbalance ratio, number of samples and overlapping level.
* SPE obtains better performance than other resampling based methods, but still worse than our proposed SCHE.
* Traditional hybrid-sampling methods fail to guarantee an improved performance by combining SMOTE and under-sampling technique. For example, SMOTETomek yields limited performance gain, or even degradtion, compared with SMOTE. This, as well as the poor performance of traditional hybrid-sampling methods when dealing with massive imbalanced data, are attributed to the inherent flaw of their strategy as we mentioned in Introduction.

TABLE III: GENERALIZED PERFORMANCE (AUPRC) ON 6 REAL-WORLD SMALL-SCALED DATASETS.

Dataset	#Samples	IR	Model	SMOTE	TLinks	SMOTEENN	SMOTETomek	SPE_{50}	$SCHE_{50}$
Coil_2000	9822	16:1	$AdaBoost_{10}$	$0.109_{\pm 0.008}$	$0.133_{\pm 0.000}$	$0.123_{\pm 0.009}$	$0.109_{\pm 0.008}$	$\mathbf{0.142}_{\pm 0.007}$	$\mathbf{0.142}_{\pm 0.013}$
			KNN	$0.089_{\pm 0.001}$	$0.088_{\pm 0.000}$	$0.093_{\pm 0.002}$	$0.089_{\pm 0.000}$	$0.113_{\pm 0.004}$	$\mathbf{0.116}_{\pm 0.005}$
Letter_img	20000	16:1	SVC	$0.998_{\pm 0.000}$	$0.998_{\pm 0.000}$	$0.997_{\pm 0.000}$	$0.998_{\pm 0.000}$	$0.998_{\pm 0.000}$	$\mathbf{0.999}_{\pm 0.000}$
Abalone_19	4177	10:1	LR	$0.023_{\pm 0.000}$	$0.018_{\pm 0.000}$	$0.023_{\pm 0.000}$	$0.023_{\pm 0.000}$	$0.017_{\pm 0.000}$	$\mathbf{0.027}_{\pm 0.004}$
			DT	$0.021_{\pm 0.008}$	$0.008_{\pm 0.000}$	$0.016_{\pm 0.004}$	$0.022_{\pm 0.008}$	$0.097_{\pm 0.051}$	$\mathbf{0.144}_{\pm 0.071}$
Webpage	34780	300:1		$0.290_{\pm 0.018}$	$0.467_{\pm 0.011}$	$0.234_{\pm 0.007}$	$0.290_{\pm 0.018}$	$0.397_{\pm 0.008}$	$\mathbf{0.675}_{\pm 0.007}$
Thyroid_sick	3772	15:1	RF_{10}	$0.904_{\pm 0.014}$	$0.922_{\pm 0.026}$	$0.904_{\pm 0.022}$	$0.897_{\pm 0.031}$	$\mathbf{0.968}_{\pm 0.004}$	$\mathbf{0.968}_{\pm 0.004}$
Optical_digits	5620	64:1	$GBDT_{10}$	$0.893_{\pm 0.006}$	$0.815_{\pm 0.000}$	$0.890_{\pm 0.007}$	$0.893_{\pm 0.006}$	$\mathbf{0.969}_{\pm 0.001}$	$\mathbf{0.969}_{\pm 0.001}$

TABLE IV: GENERALIZED PERFORMANCE ON 5 REAL-WORLD MASSIVE HIGHLY IMBALANCED DATASETS.

Dataset	Model	Metrics	SMOTE	TLinks	SMOTEENN	SMOTETomek	SPE_{50}	$SCHE_{50}$
Credit Fraud	LR	AUPRC	$0.729_{\pm 0.000}$	$0.750_{\pm 0.000}$	$0.724_{\pm 0.000}$	$0.729_{\pm 0.000}$	$0.764_{\pm 0.001}$	$\mathbf{0.773}_{\pm 0.002}$
		F1	$0.779_{\pm 0.002}$	$0.765_{\pm 0.000}$	$0.781_{\pm 0.000}$	$0.779_{\pm 0.002}$	$0.796_{\pm 0.005}$	$\mathbf{0.823}_{\pm 0.003}$
		GM	$0.781_{\pm 0.002}$	$0.765_{\pm 0.000}$	$0.782_{\pm 0.000}$	$0.781_{\pm 0.002}$	$0.800_{\pm 0.005}$	$\mathbf{0.824}_{\pm 0.003}$
		MCC	$0.695_{\pm 0.003}$	$0.765_{\pm 0.000}$	$0.692_{\pm 0.004}$	$0.695_{\pm 0.003}$	$0.797_{\pm 0.004}$	$\mathbf{0.822}_{\pm 0.002}$
	DT	AUPRC	$0.308_{\pm 0.023}$	$0.613_{\pm 0.021}$	$0.303_{\pm 0.018}$	$0.308_{\pm 0.023}$	$0.812_{\pm 0.015}$	$\mathbf{0.825}_{\pm 0.015}$
		F1	$0.521_{\pm 0.025}$	$0.782_{\pm 0.013}$	$0.515_{\pm 0.018}$	$0.521_{\pm 0.025}$	$0.850_{\pm 0.006}$	$\mathbf{0.868}_{\pm 0.003}$
		GM	$0.554_{\pm 0.021}$	$0.782_{\pm 0.013}$	$0.550_{\pm 0.017}$	$0.554_{\pm 0.021}$	$0.853_{\pm 0.006}$	$\mathbf{0.870}_{\pm 0.003}$
		MCC	$0.553_{\pm 0.021}$	$0.782_{\pm 0.013}$	$0.549_{\pm 0.017}$	$0.553_{\pm 0.021}$	$0.853_{\pm 0.006}$	$\mathbf{0.870}_{\pm 0.003}$
	KNN	AUPRC	$0.618_{\pm 0.004}$	$0.840_{\pm 0.000}$	$0.527_{\pm 0.003}$	$0.618_{\pm 0.004}$	$0.792_{\pm 0.002}$	$\mathbf{0.793}_{\pm 0.002}$
		F1	$0.770_{\pm 0.003}$	$0.778_{\pm 0.000}$	$0.713_{\pm 0.003}$	$0.770_{\pm 0.003}$	$0.848_{\pm 0.005}$	$\mathbf{0.852}_{\pm 0.002}$
		GM	$0.771_{\pm 0.002}$	$0.842_{\pm 0.000}$	$0.721_{\pm 0.002}$	$0.771_{\pm 0.002}$	$0.849_{\pm 0.005}$	$\mathbf{0.854}_{\pm 0.002}$
		MCC	$0.771_{\pm 0.002}$	$0.841_{\pm 0.000}$	$0.720_{\pm 0.002}$	$0.771_{\pm 0.002}$	$0.847_{\pm 0.005}$	$\mathbf{0.854}_{\pm 0.002}$
KDDCUP (DOS vs. R2L)		AUPRC	$0.996_{\pm 0.000}$	$0.996_{\pm 0.000}$	$0.996_{\pm 0.000}$	$0.996_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$
		F1	$\mathbf{0.998}_{\pm 0.001}$	$0.996_{\pm 0.000}$	$\mathbf{0.998}_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.001}$	$0.997_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.000}$
		GM	$\mathbf{0.998}_{\pm 0.001}$	$0.996_{\pm 0.000}$	$\mathbf{0.998}_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.001}$	$0.997_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.000}$
		MCC	$\mathbf{0.998}_{\pm 0.001}$	$0.996_{\pm 0.000}$	$\mathbf{0.998}_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.001}$	$0.997_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.000}$
KDDCUP (DOS vs. PRB)	$AdaBoost_{10}$	AUPRC	$0.998_{\pm 0.000}$	$0.999_{\pm 0.000}$	$0.998_{\pm 0.000}$	$0.998_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$
		F1	$0.986_{\pm 0.006}$	$0.990_{\pm 0.000}$	$0.989_{\pm 0.003}$	$0.986_{\pm 0.006}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.000}$
		GM	$0.986_{\pm 0.005}$	$0.990_{\pm 0.000}$	$0.989_{\pm 0.003}$	$0.986_{\pm 0.005}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.000}$
		MCC	$0.986_{\pm 0.005}$	$0.990_{\pm 0.000}$	$0.988_{\pm 0.003}$	$0.986_{\pm 0.005}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.000}$
Protein Homology	RF_{10}	AUPRC	$0.862_{\pm 0.006}$	$0.838_{\pm 0.006}$	$0.854_{\pm 0.001}$	$0.862_{\pm 0.006}$	$\mathbf{0.922}_{\pm 0.002}$	$\mathbf{0.922}_{\pm 0.001}$
		F1	$0.835_{\pm 0.011}$	$0.854_{\pm 0.006}$	$0.833_{\pm 0.007}$	$0.835_{\pm 0.011}$	$0.883_{\pm 0.002}$	$\mathbf{0.886}_{\pm 0.004}$
		GM	$0.837_{\pm 0.010}$	$0.857_{\pm 0.005}$	$0.835_{\pm 0.007}$	$0.837_{\pm 0.010}$	$0.884_{\pm 0.003}$	$\mathbf{0.889}_{\pm 0.004}$
		MCC	$0.836_{\pm 0.010}$	$0.856_{\pm 0.005}$	$0.833_{\pm 0.007}$	$0.836_{\pm 0.010}$	$0.883_{\pm 0.003}$	$\mathbf{0.887}_{\pm 0.004}$
Payment Simulation		AUPRC	$0.920_{\pm 0.009}$	$0.869_{\pm 0.007}$	$0.907_{\pm 0.014}$	$0.917_{\pm 0.009}$	$\mathbf{0.929}_{\pm 0.005}$	$0.928_{\pm 0.002}$
		F1	$0.863_{\pm 0.006}$	$0.855_{\pm 0.002}$	$0.855_{\pm 0.007}$	$0.864_{\pm 0.008}$	$0.864_{\pm 0.006}$	$\mathbf{0.868}_{\pm 0.002}$
		GM	$0.864_{\pm 0.005}$	$0.860_{\pm 0.002}$	$0.856_{\pm 0.007}$	$0.866_{\pm 0.008}$	$0.869_{\pm 0.005}$	$\mathbf{0.871}_{\pm 0.001}$
		MCC	$0.863_{\pm 0.005}$	$0.860_{\pm 0.002}$	$0.854_{\pm 0.007}$	$0.864_{\pm 0.008}$	$0.869_{\pm 0.008}$	$\mathbf{0.871}_{\pm 0.001}$

D. Effectiveness on Massive Highly Imbalanced Data

We use 5 real-world large datasets to validate SCHE's effectiveness on massive highly imbalanced classification tasks. TABLE V lists the statistics information of each dataset.

- Protein Homology[2] contains imbalanced data samples of protein sequences.
- Credit Fraud and Payment Simulation contain massive highly imbalanced transaction records. The former is based on credit card transactions and the latter on simulated online transactions.
- KDDCUP-99 is a benchmark dataset in the field of intrusion detection which contains 4 classes of attacks: R2L, PROBE, DOS, U2R. We combine the majority class (i.e., DOS) with minority class (i.e., R2L and PROBE) to construct binary classification datasets.

1) Experiment Setup: We extend the previous experiments on small-scale datasets to real-world, massive high imbalanced datasets. Our propose SCHE is first compared with 5 baselines (SMOTE, TLinks, SMOTEENN, SMOTETomek and SPE). 5 canonical classifiers (LR, DT, KNN, $AdaBoost_{10}$, RF_{10}) are used to cooperate with imbalance learning methods. Moreover, we introduce other representative resampling methods (listed in IV-D3) and ensemble methods (listed in IV-D4) for a further comprehensive comparison.

TABLE V: STATISTICS OF 5 REAL-WORLD MASSIVE HIGHLY IMBALANCED DATASETS

Dataset	#Attribute	#Sample	IR	Feature Format
Protein Homology	74	145,751	11:1	Numerical
Credit Fraud	31	284,807	578.88:1	Numerical
KDDCUP (DOS vs. R2L)	42	3,884,496	94.48:1	Integer Categorical
KDDCUP (DOS vs. PRB)	42	3,924,472	3448.82:1	Integer Categorical
Payment Simulation	11	6,362,620	773.70:1	Numerical Categorical

2) Results on Massive Highly Imbalanced Datasets: TABLE IV lists the experimental results (AUPRC, F1, G-mean

[2]Due to the massive amount of data, we put the experimental results of Protein Homology in this section.

and MCC) of proposed SCHE and 5 baseline methods. The results show that:

* SCHE demonstrates superior performance over 4 metrics with 5 used classifiers (though slightly worse than the AUPRC result of SPE on Payment Simulation).
* When dealing Credit Fraud with DT, SMOTETomek yields a huge performance degradation compared with TLinks, which is the under-sampling technique it utilizes for noise reduction. This is the same as we observed in the experimental results on real-world small-scale datasets in IV-C2 that traditional hybrid-sampling methods fail to stably boost model's performance by combining over-sampling and under-sampling technique. When we extend the previous experiments on small-scale datasets to massive highly imbalanced datasets, this instability is further manifested as performance degradation.
* SMOTETomek and SMOTE have nearly the same performance on four metrics. This may be counter-intuitive, but still makes sense. A reasonable explanation for this is that SMOTE introduces a massive influx of synthetic minority samples, which eliminates all Tomek Links in the training set, leading to the ineffectiveness of TLinks.

3) Comparison with Resampling based Methods: To fairly compare each resampling method's performance, we use Credit Fraud for evaluation because all of its features are numerical. This avoids the possible influences caused manual transformation of categorical features to numerical features. Following representative resampling methods are introduced:

- ADASYN (*ADAptive SYNthetic over-sampling*) [9] is a variant of SMOTE [8] that generates different number of samples depending on an estimate of the data distribution.
- BSMOTE (*Borderline SMOTE*) [10] is also a variant of SMOTE [8] that identifies and uses borderline samples to generate new synthetic samples.
- ROS (*Random Over-Sampling*) generates new samples by randomly repeating the current available samples.
- ENN (*Edited Nearest Neighbors*) [18] under-samples the majority class by removing samples close to the decision boundary.
- OSS (*One Sided Selection*) [42] uses TLinks to iteratively identify and remove noise samples.
- RUS (*Random Under-Sampling*) randomly selects a subset of majority class to construct a balanced training set.

TABLE VI demonstrates the experimental results of SCHE and each resampling base method. The results show that:

* Compared with resampling based methods, SCHE achieves a better performance with fewer samples and acceptable resampling time. The amount of minority samples SCHE generates depends on the difference of each cluster's size in minority class. This amount is much smaller than the difference of majority and minority class.
* Resampling based methods do achieve satisfactory performance on certain "strong" classifier like RF_{10}, but they fail to perform well when cooperating with other classifiers like DT and GBDT. In comparison, SCHE stably achieves good performance on 5 used base classifiers.
* SMOTEENN and SMOTETomek demonstrate poor performance on "weak" classifiers like DT and KNN. They also have worse performance compared with SMOTE and their respectively utilized under-sampling methods (ENN and TLinks). This means that following the design concept of traditional hybrid-sampling methods, there's no guaranteed performance gain from combining SMOTE and under-sampling technique when dealing with massive highly imbalanced datasets.
* Further more, the size of training sets that SMOTETomek and SMOTE construct are exactly the same, which means that TLinks fail to identify and remove any noise samples. As for SMOTEENN, the denoised training set is almost the same size as the one that SMOTE constructs. These observations further affirm our viewpoint that traditional hybrid-sampling methods fail to effectively reduce noise samples when dealing with massive highly imbalanced data.

4) Comparison with Ensemble Methods: Note that SCHE is an ensemble imbalanced learning framework. To further demonstrate SCHE's superior, we introduce several ensemble imbalanced learning methods for comparison:

- RUSBoost [25] integrates RUS in the learning of AdaBoost to obtain balanced training sets.
- Cascade [26] iteratively drops majority samples that have been well-learned by the ensemble.
- Easy (*EasyEnsemble*) [26] is an ensemble of AdaBoost trained on balanced bootstrap samples obtained by RUS.
- DAPS (*DynAmic self-Paced Sampling ensemble*) [27] identifies 'vetoed' samples and assign different weights according to their learning difficulty.
- SMOTEBagging [20] is a bagging [22] ensemble that applies SMOTE [8] to balance the dataset in each iteration.
- SMOTEBoost [21] is a boosting [23] ensemble that applies SMOTE [8] to balance the dataset in each iteration.

We apply all methods on Credit Fraud and adopt DT as the base classifier for ensemble imbalanced learning methods. Since the number of base classifiers is a key hyper-parameter of ensemble model, we decrease the value of this parameter from 50 to 20 and 15 to validate SCHE's effectiveness when working with fewer base classifiers. Corresponding results are demonstrated in TABLE VII. Due to space limitation, the results of Cascade, RUSBoost and Easy are not listed because they perform clearly worse than other baselines.

* SCHE outperforms 5 ensemble baselines. When working with fewer base classifiers, SCHE still demonstrates effectiveness of stably boosting base classifier's performance.
* SCHE obtains a trade-off between model performance and training samples amount. On the on hand, SCHE requires fewer training samples than over-sampling based ensemble methods to obtain a better performance. On the other hand, SCHE outperform under-sampling based

TABLE VI: GENERALIZED PERFORMANCE (AUPRC) OF RESAMPLING METHODS

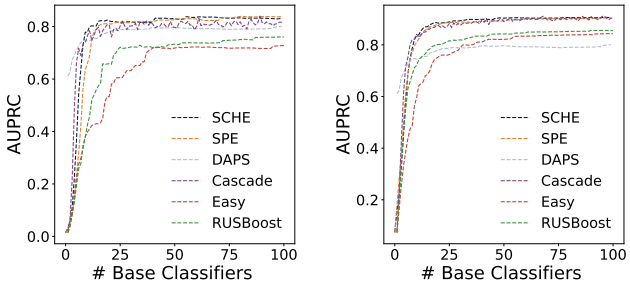
Category	Over-sampling				Under-sampling				Hybrid-sampling		Ensemble
Model \ Method	ADASYN	BSMOTE	SMOTE	ROS	ENN	OSS	RUS	TLinks	SMOTEENN	SMOTETomek	SCHE_{50}
LR	$0.707_{\pm 0.000}$	$0.710_{\pm 0.000}$	$0.729_{\pm 0.000}$	$0.742_{\pm 0.000}$	$0.761_{\pm 0.000}$	$0.750_{\pm 0.000}$	$0.599_{\pm 0.115}$	$0.750_{\pm 0.000}$	$0.724_{\pm 0.001}$	$0.729_{\pm 0.000}$	$\mathbf{0.773}_{\pm 0.002}$
DT	$0.304_{\pm 0.023}$	$0.521_{\pm 0.019}$	$0.308_{\pm 0.024}$	$0.557_{\pm 0.014}$	$0.601_{\pm 0.023}$	$0.610_{\pm 0.023}$	$0.014_{\pm 0.001}$	$0.613_{\pm 0.021}$	$0.303_{\pm 0.018}$	$0.308_{\pm 0.023}$	$\mathbf{0.825}_{\pm 0.015}$
KNN	$0.610_{\pm 0.003}$	$0.730_{\pm 0.000}$	$0.618_{\pm 0.004}$	$0.706_{\pm 0.000}$	$0.721_{\pm 0.000}$	$0.778_{\pm 0.000}$	$0.336_{\pm 0.096}$	$0.778_{\pm 0.000}$	$0.527_{\pm 0.003}$	$0.618_{\pm 0.004}$	$\mathbf{0.793}_{\pm 0.002}$
GBDT$_{10}$	$0.354_{\pm 0.028}$	$0.226_{\pm 0.009}$	$0.703_{\pm 0.004}$	$0.693_{\pm 0.025}$	$0.706_{\pm 0.000}$	$0.535_{\pm 0.154}$	$0.331_{\pm 0.206}$	$0.658_{\pm 0.008}$	$0.702_{\pm 0.004}$	$0.703_{\pm 0.004}$	$\mathbf{0.820}_{\pm 0.004}$
RF$_{10}$	$0.806_{\pm 0.004}$	$0.807_{\pm 0.016}$	$0.798_{\pm 0.013}$	$0.798_{\pm 0.012}$	$0.762_{\pm 0.016}$	$0.803_{\pm 0.015}$	$0.497_{\pm 0.080}$	$0.797_{\pm 0.007}$	$0.782_{\pm 0.010}$	$0.798_{\pm 0.013}$	$\mathbf{0.854}_{\pm 0.003}$
#Sample	454937	454904	454904	454904	227714	215551	786	227824	454548	454904	88050
Re-sampling Time(s)	0.23	0.24	0.26	0.06	19.45	19.34	0.03	23.20	131.72	100.02	13.99×50

TABLE VII: GENERALIZED PERFORMANCE OF ENSEMBLE METHODS

Category		Over-sampling		Under-sampling			Hybrid-sampling
# Base Classifiers	**Metrics**	$\texttt{SMOTEBagging}_n$	$\texttt{SMOTEBoost}_n$	$\texttt{Cascade}_n$	$\texttt{SPE}_n$	$\texttt{DAPS}_n$	$\texttt{SCHE}_n$
n=50	AUPRC	$0.799_{\pm 0.011}$	$0.435_{\pm 0.019}$	$0.805_{\pm 0.010}$	$0.812_{\pm 0.015}$	$0.806_{\pm 0.008}$	$\mathbf{0.825}_{\pm 0.015}$
	F1	$0.865_{\pm 0.005}$	$0.642_{\pm 0.016}$	$0.844_{\pm 0.009}$	$0.850_{\pm 0.006}$	$0.848_{\pm 0.005}$	$\mathbf{0.868}_{\pm 0.003}$
	GM	$0.867_{\pm 0.004}$	$0.651_{\pm 0.015}$	$0.845_{\pm 0.009}$	$0.853_{\pm 0.006}$	$0.849_{\pm 0.005}$	$\mathbf{0.870}_{\pm 0.003}$
	MCC	$0.866_{\pm 0.004}$	$0.582_{\pm 0.019}$	$0.845_{\pm 0.009}$	$0.853_{\pm 0.006}$	$0.849_{\pm 0.005}$	$\mathbf{0.870}_{\pm 0.003}$
	#Sample	22,745,200	22,745,200	39,300	39,300	39,300	88,050
	Resampling Time(s)	542.13	1421.15	1.86	2.77	177.98	29.04
n=20	AUPRC	$0.787_{\pm 0.015}$	$0.432_{\pm 0.019}$	$0.785_{\pm 0.015}$	$0.795_{\pm 0.011}$	$\mathbf{0.800}_{\pm 0.010}$	$0.799_{\pm 0.013}$
	F1	$0.861_{\pm 0.007}$	$0.640_{\pm 0.018}$	$0.834_{\pm 0.011}$	$0.847_{\pm 0.007}$	$0.856_{\pm 0.005}$	$\mathbf{0.865}_{\pm 0.007}$
	GM	$0.863_{\pm 0.007}$	$0.650_{\pm 0.016}$	$0.835_{\pm 0.012}$	$0.850_{\pm 0.007}$	$0.857_{\pm 0.005}$	$\mathbf{0.867}_{\pm 0.007}$
	MCC	$0.863_{\pm 0.006}$	$0.581_{\pm 0.016}$	$0.835_{\pm 0.012}$	$0.850_{\pm 0.006}$	$0.857_{\pm 0.005}$	$\mathbf{0.867}_{\pm 0.007}$
	#Sample	9,098,080	9,098,080	15,720	15,720	15,720	35,220
	Resampling Time(s)	202.86	626.60	0.77	1.12	94.69	26.61
n=15	AUPRC	$0.783_{\pm 0.011}$	$0.436_{\pm 0.020}$	$0.782_{\pm 0.012}$	$0.784_{\pm 0.013}$	$\mathbf{0.794}_{\pm 0.018}$	$\mathbf{0.794}_{\pm 0.011}$
	F1	$0.857_{\pm 0.004}$	$0.643_{\pm 0.0179}$	$0.839_{\pm 0.014}$	$0.841_{\pm 0.007}$	$0.856_{\pm 0.004}$	$\mathbf{0.861}_{\pm 0.006}$
	GM	$0.859_{\pm 0.004}$	$0.652_{\pm 0.016}$	$0.841_{\pm 0.014}$	$0.843_{\pm 0.006}$	$0.857_{\pm 0.003}$	$\mathbf{0.863}_{\pm 0.006}$
	MCC	$0.859_{\pm 0.004}$	$0.576_{\pm 0.018}$	$0.841_{\pm 0.014}$	$0.843_{\pm 0.006}$	$0.857_{\pm 0.003}$	$\mathbf{0.863}_{\pm 0.006}$
	#Sample	6,823,560	6,823,560	11,790	11,790	11,790	26,415
	Resampling Time(s)	149.65	412.20	0.58	0.80	83.67	26.29

ensemble methods by generating an acceptable amount of minority samples

We further apply SCHE and 5 under-sampling based ensemble baselines (SPE, DAPS, Cascade, Easy and RUSBoost) with the number of base classifiers ranging from 1 to 100 on Credit Fraud and Protein Homology. DT is used as base classifier. Due to huge computation cost, SMOTEBagging and SMOTEBoost are neglected. We demonstrate each method's AUPRC variation within the increasing number of base classifiers in Fig. 7. SCHE has a higher performance upper bound as well as a smoother performance growth curve. Cascade, RUSBoost and Easy do not demonstrate a steady performance growth as the number of base classifiers increases. DAPS utilizes weighting mechanism and obtains high performance with very few base classifiers. But it has a relatively lower performance upper bound.



(a) Credit Fraud (b) Protein Homology

Fig. 7: AUPRC variation of SCHE and 5 under-sampling based ensemble baselines with number of base classifiers ranging from 1 to 100 on Credit Fraud and Protein Homology.

E. Extensive Study on Instance Hardness & Hyper-parameters

1) Effectiveness of Instance Hardness: Since the training process of SCHE is controlled by instance hardness, which is a weighted combination of dynamic and local hardness, we are interested in how SCHE performs when these 3 hardness independently guides the training process.

We set d_weight_i and l_weight_i in Eq. 8 to 1 to control the sampling process by dynamic hardness (SCHE(dh)) and local hardness (SCHE(lh)). Our proposed SCHE, which utilizes instance hardness, is notated as SCHE(dh+lh). Fig. 8 shows the AUPRC results on CheckerBoard, Credit Fraud and Protein Homology (results of SMOTEENN and SMOTETomek are demonstrated for a clearer comparison).

Fig. 8 shows that with the weighted combination of dynamic and local hardness, SCHE(dh+lh) achieves best performance. This is because instance hardness comprehensively reflects samples' learning difficulty from data/model aspects.

2) Sensitivity to Hyper-parameters k: In SCHE, the value of k is an important parameter used to:

- Calculate local hardness for minority samples in Eq. 2.

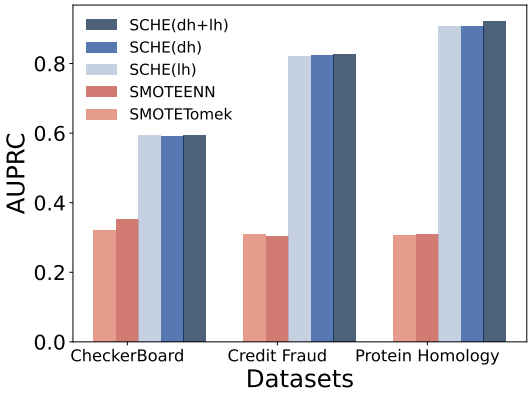


Fig. 8: AUPRC performance of SCHE when utilizing local, dynamic and instance hardness respectively.

- Determine the number of nearest neighbors when generating new samples according to Eq. 10.
- Determine the *minPts* parameter of DBSCAN. *minPts* is the lower bound for the size of clusters DBSCAN defines.

We apply SCHE_{50} on Credit Fraud with k ranging from 1 to 40. Fig. 9 demonstrates the mean of AUPRC in 10 independent runs with different k values. SCHE is naturally sensitive to k as we expected because it over-samples the minority class based on k-nearest neighbor and local hardness.

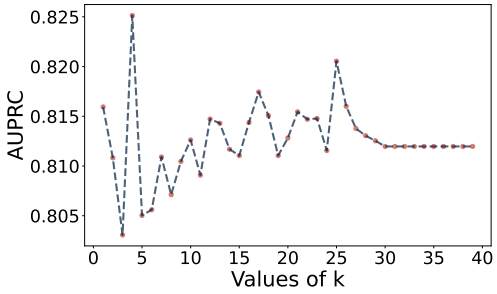


Fig. 9: Sensitivity to hyper-parameter k.

3) Selection of ϵ in DBSCAN: The effectiveness of SCHE depends on the appropriate hyper-parameters of DBSCAN (*minPts* and ϵ). This is because SCHE conducts cluster-wise over-sampling and calculates the amount of generated minority samples by summing up the difference of each cluster's size, which both require a well-defined cluster structure. In our experiments, *minPts* is the same value as k.

Aiming at tapping the potential of cluster-wise over-sampling, we want to maximize the number of clusters obtained by DBSCAN to generate more minority samples within a reasonable range. We utilize 3 well-known metrics for clustering evaluation[3], DBI(*Davies-Bouldin Index*) [43], SC (*Silhouette Coefficient*) [44], CHI (*Calinski-Harabasz Index*) [45] to help select the optimal ϵ. A lower DBI or higher SC and CHI relate to a better defined clustering result. To

[3]Due to space limitation, we only illustrate the correlation between quality of clustering result and these 3 metrics instead of their specific formulations.

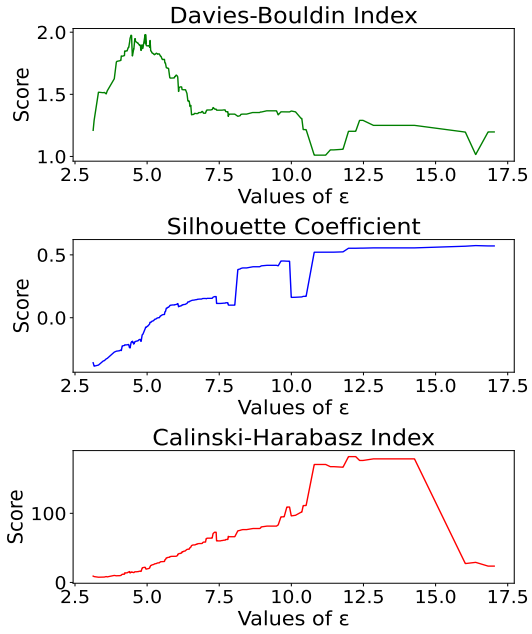


Fig. 10: Metric scores of clustering results obtain by DBSCAN with different ϵ.

determine the optimal ϵ, we calculate the distance between a sample and its k^{th} neighbor of each minority samples and sort these distances in ascending order. Then, we set ϵ to the value of these distances and conduct DBSCAN. Finally, we obtain DBI, SC and CHI scores of clustering results with various ϵ.

Take Credit Fraud for example, Fig. 10 demonstrates the metric scores with varying ϵ on its minority class. From the correlation of 3 metrics with the quality of the clustering results, we tend to select ϵ wih low DBI as well as high SC and CHI. Hence, we compare the number of clusters w.r.t ϵ ranging from 10.0 to 15.0 and determine 10.3 as the value of ϵ because DBSCAN gets the most clusters using this ϵ value.

V. Conclusion

In this paper, we analyze the inherent flaw of traditional hybrid-sampling methods when dealing with massive highly imbalanced data. We argue that generating a large amount of samples to handle imbalanced data under such scenario is impractical and prove this viewpoint through detailed experiments. We proposed SCHE, a *Self-paced Curriculum Hybrid-sampling based Ensemble* to learn from massive highly imbalanced data. Compared with other methods, SCHE achieves superior performance with acceptable computation cost.

We must acknowledge the limitation of SCHE that its over-sampling phase relies on manually determined cluster structure, thereby constraining its applicability. Future work can investigate how to automatically determine cluster structure in a close combination with hybrid-sampling process in ensemble imbalanced learning framework.

References

[1] B. Ashmore and L. Chen, "Hover: Homophilic oversampling via edge removal for class-imbalanced bot detection on graphs," in *Proceedings of*

the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 3728–3732.

[2] Z. Ren, T. Lin, K. Feng, Y. Zhu, Z. Liu, and K. Yan, "A systematic review on imbalanced learning methods in intelligent fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[3] S. Susan and A. Kumar, "The balancing trick: Optimized sampling of imbalanced datasets—a brief survey of the recent state of the art," *Engineering Reports*, vol. 3, no. 4, p. e12298, 2021.

[4] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[5] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[6] F. Cheng, J. Zhang, and C. Wen, "Cost-sensitive large margin distribution machine for classification of imbalanced data," *Pattern Recognition Letters*, vol. 80, pp. 107–112, 2016.

[7] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on knowledge and data engineering*, vol. 18, no. 1, pp. 63–77, 2005.

[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*, vol. 16, pp. 321–357, 2002.

[9] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 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," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[11] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[12] K. Yang, Z. Yu, X. Wen, W. Cao, C. P. Chen, H.-S. Wong, and J. You, "Hybrid classifier ensemble for imbalanced data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 4, pp. 1387–1400, 2019.

[13] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[14] G. E. Batista, A. L. Bazzan, M. C. Monard *et al.*, "Balancing training data for automated annotation of keywords: a case study." *Wob*, vol. 3, pp. 10–8, 2003.

[15] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, 2021.

[16] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.

[17] I. Tomek, "Two modifications of cnn." 1976.

[18] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[19] A. Zhang, H. Yu, Z. Huan, X. Yang, S. Zheng, and S. Gao, "Smote-rknn: A hybrid re-sampling method based on smote and reverse k-nearest neighbors," *Information Sciences*, vol. 595, pp. 70–88, 2022.

[20] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *2009 IEEE symposium on computational intelligence and data mining*. IEEE, 2009, pp. 324–331.

[21] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*. Springer, 2003, pp. 107–119.

[22] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[23] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[24] S. M. Liu, J.-H. Chen, and Z. Liu, "An empirical study of dynamic selection and random under-sampling for the class imbalance problem," *Expert Systems with Applications*, vol. 221, p. 119703, 2023.

[25] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, vol. 40, no. 1, pp. 185–197, 2009.

[26] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2008.

[27] F. Zhou, S. Gao, L. Ni, M. Pavlovski, Q. Dong, Z. Obradovic, and W. Qian, "Dynamic self-paced sampling ensemble for highly imbalanced and class-overlapped data classification," *Data Mining and Knowledge Discovery*, vol. 36, no. 5, pp. 1601–1622, 2022.

[28] Z. Liu, W. Cao, Z. Gao, J. Bian, H. Chen, Y. Chang, and T.-Y. Liu, "Self-paced ensemble for highly imbalanced massive data classification," in *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 2020, pp. 841–852.

[29] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[30] N. S. Altman, "An introduction to kernel and nearest-neighbor non-parametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[31] X. Tao, Y. Zheng, W. Chen, X. Zhang, L. Qi, Z. Fan, and S. Huang, "Svdd-based weighted oversampling technique for imbalanced and overlapped dataset learning," *Information Sciences*, vol. 588, pp. 13–51, 2022.

[32] Q. H. Doan, S.-H. Mai, Q. T. Do, and D.-K. Thai, "A cluster-based data splitting method for small sample and class imbalance problems in impact damage classification," *Applied Soft Computing*, vol. 120, p. 108628, 2022.

[33] Y. Yuan, J. Wei, H. Huang, W. Jiao, J. Wang, and H. Chen, "Review of resampling techniques for the treatment of imbalanced industrial data classification in equipment condition monitoring," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 106911, 2023.

[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[35] Y. Wang, W. Gan, J. Yang, W. Wu, and J. Yan, "Dynamic curriculum learning for imbalanced data classification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5017–5026.

[36] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 8577–8584.

[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[38] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[39] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[40] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[41] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[42] M. Kubat, S. Matwin *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, no. 1. Citeseer, 1997, p. 179.

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[44] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[45] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.